TRABAJO DE FIN DE GRADO

Universidad Carlos III de Madrid Grado en Ingeniería Informática

Sistema de interfaz genérico para la representación de información visual de modelos

Profesor: Juan Bautista Llorens Morillo

Alumno: Alberto Guzmán Aroca

23/09/2015

Índice

1. Introduction 10
1.1 Overview 10
1.2 Objectives 11
2. Estado del arte 12
2.1 Gestión del conocimiento 12
2.2 Knowledge Manager 14
2.2.1 Ontología 15
2.3 RSHIP 16
2.4 Sistema universal de edición gráfica 16
2.4.1 Librerías gráficas 17
2.4.2 Objetivo de la mejora 18
3. Estudio de viabilidad 19
3.1 Situación actual del sistema 19
3.2 Especificación de requisitos del sistema 21
3.2.1 Requisitos de capacidad 23
3.2.2 Requisitos de restricción 27
3.2.3 Requisitos inversos 28
3.3 Estudio de alternativas tecnológicas para el proyecto 29
3.3.1 Tecnologías necesarias para el desarrollo 29
3.3.2 Gestión del proyecto 32
3.3.3 Gestión de versiones 33
3.3.4 Gestión de pruebas 34
3.3.5 Gestión de tareas 34
3.4 Librerías gráficas 34
3.4.1 Lasalle Workflow 35
3.5 Representaciones para los nodos 36
3.5.1 Serialización de las imágenes como atributos de archivos XML 37
3.5.2 Serialización de imágenes mediante Base64 Strings 38
3.6 Proceso de selección de algoritmos para dibujar diagramas de circuitos eléctricos 40
3.6.1 Configuraciones de circuitos eléctricos según las necesidades del cliente 40
3.6.2 Selección de algoritmos para detectar ciclos 43
4. Análisis 46

4.1 Definición del sistema 46
4.1.1 Determinación del alcance del sistema 46
4.1.2 Identificación del Entorno Tecnológico 48
4.1.3 Especificación de estándares y normas 49
4.1.4 Identificación de los usuarios participantes y finales 50
4.2 Identificación de subsistemas de análisis 51
4.3 Establecimiento de requisitos 52
4.3.1 Obtención de requisitos del sistema 52
4.3.2 Análisis de los requisitos 61
4.3.3 Especificación de casos de uso 62
4.3.3 Matriz de trazabilidad de Requisitos de Sistema – Casos de Uso 71
4.4 Análisis de clases 73
4.4.1 Identificación de responsabilidades y atributos 73
4.5 Definición de interfaces de usuario 79
4.4.2 Especificación de principios generales de la interfaz 79
4.4.3 Especificación de formatos individuales de la interfaz de pantalla 81
4.4.4 Identificación de diálogos 85
4.5 Especificación del plan de pruebas 89
4.5.1 Definición del Alcance de las Pruebas 89
4.5.2 Definición de Requisitos del entorno de Pruebas 89
4.5.3 Definición de las Pruebas de Aceptación del Sistema 89
4.5.4 Matriz de trazabilidad entre Pruebas de aceptación y Requisitos. 96
5. Diseño 98
5.1 Definición de la arquitectura del sistema 98
5.1.1 Definición de los niveles de arquitectura 98
5.1.2 Especificación de estándares y normas de diseño y construcción 101
5.1.3 Identificación de los subsistemas de diseño 102
5.2 Diseño de clases 102
5.2.1 Identificación de clases 105
5.2.2 Especificación de clases 105
5.2.3 Especificación de mejoras sobre los scripts 117
5.3 Diseño de casos de uso reales 123
5.4 Especificación técnica del plan de pruebas 131

5.4.1 Definición del alcance de las pruebas 131
5.4.2 Pruebas unitarias 131
6. Planificación y presupuesto 136
6.1 Planificación 136
6.2 Presupuesto 141
7. Líneas futuras 144
8. Conclusions 145
8.1 Planning variances 145
8.2 Accomplished objectives 145
8.3 Personal Evaluation 145
9. Anexos 147
9.1 Glosario de términos 147
9.2 Acrónimos 147
9.3 Citas Bibliográficas 148
9.4 User Manual (Updated notes) 149
9.4.1 Introduction 149
9.4.2 Let's refresh your mind with the user interface 149
9.4.3 Layout in the native and transformation view 151
9.4.4 The new artifact types with representation and the transformation view changes 151
9.4.5 Circuit Diagram Layout 155

Índice de ilustraciones

Ilustración 1: Knowledge Manager Logo 10
Ilustración 2: Esquema del proceso de la gestión del conocimiento 13
Ilustración 3: Pantalla del módulo de representación gráfica con los elementos principales representados 17
Ilustración 4: Logotipo de la empresa detrás de la librería JavaScript Lassalle AddFlow 18
Ilustración 5: Los tres principales lenguajes de programación usados para el módulo de representación gráfica 31
Ilustración 6: Logo del gestor de tareas Trello 34
Ilustración 7: Funcionamiento y arquitectura para Lassalle 36
Ilustración 8: Ejemplo de documento XML 37
Ilustración 9: Imagen antes y después de la codificación en string de Base 64 39
Ilustración 10: Circuito eléctrico serie/paralelo sin cerrar 40

Ilustración 11: Circuito eléctrico serie/paralelo cerrado, primer caso 41
Ilustración 12: Circuito eléctrico serie/paralelo cerrado, segundo caso 42
Ilustración 13: Circuito eléctrico serie/paralelo cerrado, tercer caso 42
Ilustración 14: identificación de una componente fuertemente conexa con el Algoritmo de Kosaraju paso a paso 45
Ilustración 15: Diagrama de alcance de sistema creado por Pablo Sánchez Pérez 47
Ilustración 16: Casos de uso relacionados con la barra de herramientas 64
Ilustración 17: Casos de uso relacionados con la interacción del usuario con los elementos del canvas.. 69
Ilustración 18: Pantalla de edición en su configuración original con la barra de herramientas desplegada. 81
Ilustración 19: Pantalla de edición con modificaciones 82
Ilustración 20: Identificador de tipo de diagrama 82
Ilustración 21: Botón de añadir nuevo tipo de artefacto 82
Ilustración 22: Paleta de edición cuando se edita un circuito eléctrico 83
Ilustración 23: Ventana de transformación original 83
Ilustración 24: Pantalla de transformación con modificaciones 84
Ilustración 25: Ventana para nuevos tipos de artefacto 85
Ilustración 26: Ventana de edición de artefacto con lista desplegada para la selección del tipo de artefacto 87
Ilustración 27: Circuito eléctrico cerrado 87
Ilustración 28: Ventana de edición de artefacto dentro de un diagrama de circuito eléctrico 88
Ilustración 29: Ventana de edición de término en diagramas de circuito eléctrico 88
Ilustración 30: Arquitectura Modelo Vista Controlador 99
Ilustración 31: Niveles de arquitectura aplicados a este sistema 100
Ilustración 32: Diagrama de clases original 103
Ilustración 33: Nuevo diagrama de clases 104
Ilustración 34: Ejemplo de forma personalizada de un elemento sin representación gráfica en un diagrama que sí cuenta con elementos con representaciones 120
Ilustración 35: Ejemplo de forma personalizada de un elemento con representación gráfica 120
Ilustración 36: Diagrama de secuencia CU-001: Efectuar acción de layout, para filosofía de dibujo asociada a un diagrama de circuito eléctrico en el modo nativo 124
Ilustración 37: Diagrama de secuencia CU-002: Efectuar acción de layout, para filosofía de dibujo asociada a un diagrama de circuito eléctrico en el modo transformado. 125
Ilustración 38: Diagrama de secuencia CU-003: Efectuar acción de layout, para filosofía de dibujo asociada a un diagrama UML en el modo transformado 126
Ilustración 39: Diagrama de secuencia CU-004: Añadir nuevos tipos de artefactos 126
Ilustración 40: Diagrama de secuencia CU-005: Definir layout del diagrama 127
Ilustración 41: Diagrama de secuencia CU-006: Transformar el diagrama con elementos que tienen asociada una representación 128
Ilustración 42: Diagrama de secuencia CU-007: Recolocar los elementos del diagrama 129
Ilustración 43: Diagrama de secuencia CU-008: Modificar tipo de un artefacto 129
Ilustración 44: Diagrama de secuencia CU-009: Seleccionar primer elemento de un ciclo en circuitos eléctricos 130
Ilustración 45: Diagrama de Gantt 140

Ilustración 46: Set de iconos de Visio 2013 para diagramas de redes 144
Ilustración 47: Original User Interface 149
Ilustración 48: New and changed user interface 150
Ilustración 49: Layout action button in the transformation view 151
Ilustración 50: Icons in a diagram in the transformation view 151
Ilustración 51: New Artifact Type Steps 152
Ilustración 52: Steps in the native view to give representations to the diagram's artifacts 153
Ilustración 53: Transformed diagram with its representations 154
Ilustración 54: Changes kept between the native and transformation view 154
Ilustración 55: First Step to Circuit diagram action layout 155
Ilustración 56: Second Step to Circuit diagram action layout 156
Ilustración 57: Third Step to Circuit diagram action layout 156
Ilustración 58: Reorganization of the nodes after applying the layout action 156

Índice de tablas

Tabla 1: Ejemplo de tabla para requisitos 23
Tabla 2: RU-C01: Visualización gráfica de información de cualquier tipo. 23
Tabla 3: RU-C02: Edición de la información gráfica 23
Tabla 4: RU-C03: Creación de cualquier tipo de información gráfica 24
Tabla 5: RU-C04: Gestión de información gráfica 24
Tabla 6: RU-C05: Almacenado de la información 24
Tabla 7: RU-C06: Visualización de información transformada 25
Tabla 8: RU-C07: Exportado de la información gráfica 25
Tabla 9: RU-C08: Localización del Sistema 25
Tabla 10: RU-C09: Hacer y deshacer 25
Tabla 11: RU-C10: Layout 26
Tabla 12: RU-C11: Seleccionar tipo de Layout 26
Tabla 13: RU-C11: Layout de circuito eléctrico 26
Tabla 14: RU-C12: Cargar iconos en los nodos 26
Tabla 15: RU-C14: Cambios en el diagrama 27
Tabla 16: RU-R01: Restricción de entorno operativo 27
Tabla 17: RU-R02: Tamaño máximo de las imágenes 27
Tabla 18: RU-I01: Funcionamiento como subsistema 28
Tabla 19: RU-I02: Imágenes no almacenadas como recurso local 28
Tabla 20: Tabla con especificaciones de las alternativas para el sistema operativo 30
Tabla 21: Tabla con especificaciones de los IDEs 32
Tabla 22: Ejemplo de tabla de requisito de sistema 53
Tabla 23: RS-F001 - Crear nuevo tipo de artefacto 54
Tabla 24: RS-F002 – Editar artefacto existente para darle una nueva representación 54
Tabla 25: RS-F003- Sólo se mostrará la imagen asociada a un tipo de artefacto en el modo transformado 55
Tabla 26: RS-F004 – Los nuevos tipos de artefactos se podrán almacenar en la base de datos 55
Tabla 27: RS-F005 - Las representaciones de los nodos podrán rotar 56
Tabla 28: RS-F007 -Creación de nuevos layouts 56
Tabla 29: RS-F007: Editar layout asociado al diagrama 57
Tabla 30: RS-F008 – Límites en la selección de filosofías de dibujo – II 57
Tabla 31: RS-F009 – El sistema debe diferenciar los layouts 58
Tabla 32: RS-F010 – Acción de layout en modo nativo y transformado 58
Tabla 33: RS-F011 – Cambios en la disposición de los elementos del diagrama 59
Tabla 34: RS-F012 – Restricciones en la edición de layout de un diagrama 59
Tabla 35: RS-F013 – Rotación y traslación de las imágenes tras modificaciones del diagrama 59
Tabla 36: RS-F014 – Layout de circuito eléctrico basado en una filosofía serie-paralela 60
Tabla 37: RS-In001: Opción de añadir nuevo tipo de artefacto 60
Tabla 38: RS-In002 - Acción de layout 61
Tabla 39: RS-In003 – Opción de personalizar el layout del diagrama 61
Tabla 40: Matriz de trazabilidad entre los requisitos de usuario y de sistema 62

Tabla 41: Plantilla de ejemplo para los casos de uso 63
Tabla 42: CU-001: Efectuar acción de layout 65
Tabla 43: CU-002: Efectuar acción de layout, para filosofía de dibujo asociada a un diagrama de circuito eléctrico en el modo transformado 65
Tabla 44: CU-003: Efectuar acción de layout, para filosofía de dibujo asociada a un diagrama UML en el modo transformado 66
Tabla 45: CU-004: Añadir nuevos tipos de artefactos 67
Tabla 46: CU-005: Definir layout del diagrama 67
Tabla 47: CU-006: Transformar el diagrama con elementos que tienen asociada una representación 68
Tabla 48: CU-007: Recolocar los elementos del diagrama 70
Tabla 49: CU-008: Modificar tipo de un artefacto. 70
Tabla 50: CU-009: Seleccionar primer elemento de un ciclo en circuitos eléctricos. 71
Tabla 51: Matriz de trazabilidad de Requisitos de Sistema – Casos de Uso 72
Tabla 52: Plantilla usada para la descripción de las clases 73
Tabla 53: CL-01: Propiedades del circuito 74
Tabla 54: CL-02: Propiedades generales del nodo 74
Tabla 55: CL-03: Tipo de representación 75
Tabla 56: CL-04: Gráfico 75
Tabla 57: CL-05: Relación 75
Tabla 58: CL-06: Nodo 76
Tabla 59: CL-07: Formulario para crear nuevo tipo de artefacto 76
Tabla 60: CL-08: Formulario para editar artefacto del diagrama 77
Tabla 61: CL-09: Formulario para editar término del diagrama 77
Tabla 62: CL-10: Formulario para editar el tipo de artefacto asignado al diagrama 77
Tabla 63: CL-11: Mejoras en graphicalArtifactUserControl 78
Tabla 64: CL-12: Mejoras en Transformation 78
Tabla 65: Principios seguidos el año pasado para definir la interfaz 80
Tabla 66: Plantilla para las tablas de pruebas de aceptación de sistema 90
Tabla 67: PA-01: Comprobar edición de layout asociado a un diagrama 90
Tabla 68: PA-02: Comprobar acción de layout para circuito eléctrico en modo nativo 91
Tabla 69: PA-03: Comprobar acción de layout para circuito eléctrico en modo transformado 91
Tabla 70: PA-04: Comprobar edición de layout asociado a un diagrama sólo en modo nativo 92
Tabla 71: PA-05: Comprobar que se conservan los cambios desde el modo nativo al transformado 92
Tabla 72: PA-06: Comprobar que se conservan los cambios desde el modo transformado al nativo 92
Tabla 73: PA-07: Comprobar que se añade correctamente un nuevo tipo de artefacto 93
Tabla 74: PA-08: Editar artefacto para asignarle un tipo de artefacto con representación y comprobar que se muestra la misma 93
Tabla 75: PA-09: Comprobar que las imágenes de los artefactos rotan en el modo transformado tras aplicar layout 94
Tabla 76: PA-10: Comprobar que las imágenes de los artefactos rotan en el modo transformado cuando el usuario mueve los nodos 95
Tabla 77: PA-11: Comprobar campos obligatorios de un nuevo tipo de artefacto 96
Tabla 78: Matriz de trazabilidad entre Pruebas de aceptación y Requisitos 97
Tabla 79: CLD-01: CircuitProperties 106

Tabla 80: CLD-02: PropertiesGeneral 106
Tabla 81: CLD-03: RepresentationType 107
Tabla 82: CLD-04: Graph 108
Tabla 83: CLD-05: Relation 109
Tabla 84: CLD-06: Node 109
Tabla 85: CLD-07: NewArtifactTypeForm 110
Tabla 86: CLD-08: ArtifactForm 111
Tabla 87: CLD-09: TermForm 111
Tabla 88: CLD-10: TypeArtifactForm 112
Tabla 89: CLD-11: graphicalArtifactUserControl 114
Tabla 90: CLD-12: Transformation 116
Tabla 91: CLJ-01: addflow 118
Tabla 92: CLJ-02: layoutflow 120
Tabla 93: CLJ-03: customShapes 120
Tabla 94: CLJ-04: manager 121
Tabla 95: CLJ-05: toolBar 122
Tabla 96: Plantilla para la tabla de pruebas unitarias del sistema 131
Tabla 97: PU- 01: Comprobar nueva transformación con nodos con representación 132
Tabla 98: PU- 02: Comprobar primer nodo de diagramas de circuito eléctrico 132
Tabla 99: PU- 03: Comprobar primer nodo de una componente fuertemente conexa. 133
Tabla 100: PU- 04: Comprobar que se guardan en los nodos la propiedades necesarias para diagramas de circuitos eléctricos 133
Tabla 101: PU- 05: Comprobar que se devuelve la primera relación de un ciclo en un diagrama de circuito eléctrico 134
Tabla 102: PU- 06: Comprobar que se devuelve la última relación de un ciclo en un diagrama de circuito eléctrico 134
Tabla 103: PU- 07: Comprobar el ángulo que forman dos puntos para el giro de las imágenes 135
Tabla 104: PU- 08: Comprobar el ángulo de giro de la representación de un nodo en la transformación del diagrama 135
Tabla 105: Estimación de horas 137
Tabla 106: Horas totales por jornadas laborables en los meses trabajados 137
Tabla 107: Estimación de esfuerzo para el mes de febrero 138
Tabla 108: Estimación de esfuerzo para el mes de marzo 138
Tabla 109: Estimación de esfuerzo para el mes de abril 139
Tabla 110: Estimación de esfuerzo para el mes de mayo 139
Tabla 111: Estimación de esfuerzo para el mes de junio 140
Tabla 112: Coste de material y transporte 141
Tabla 113: Sueldos anuales y a la hora por los roles asumidos en este proyecto. 142
Tabla 114: Coste total y por rol por esfuerzo en cada tarea 142
Tabla 115: Desglose completo del presupuesto 143

1. Introduction

In this introductory chapter, it is intended to show a global vision of this project, the main theme, identifying its requirements and its goal.

1.1 Overview

To date Software engineering has become one of the most powerful major pillars at business environment.
Its competences are widely practiced at ITIL, information technologies and on any other technologic development.

Therefore, this project's objective should accomplish a customer need, but taking into account the best technical design, construction and maintenance software, and taking care of the application environment and customer requirements.

This project is a modification to improve another existing project. This existing project is a graphical visual representation of information module, which belong to the Knowledge Manager (KM) application of the The Reuse Company.

Ilustración 1: Knowledge Manager Logo

As a brief description of this application, the KM is a knowledge management tool, used as requirement management and analyzer as an example.
However, its main capacity is to give interoperability between different types of information from an organization.

Interoperability defined by IEEE as:

"The capacity of two or more systems or components to exchange information and use the information exchanged."

How does the KM get this interoperability?

Thanks to the KM's representation model of information, RSHIP. It will be detailed in the state of art chapter.

The KM's graphical visual representation of information module stems from the need of giving to an organization a quick view of the information saved in the KM. So this module pretends to illustrate any kind of information contained in the RSHIP model.

However, can this module illustrate all the information graphically?

The answer is no and this is the project´s objective. With the development of some improvements over the module during this project, it is intended to give this module the capacity of represent more information types, as for example, circuit diagrams.

Therefore, the requirements for this project are to improve the functionality of this tool adding the circuit diagram representation and adding images to the elements of these diagrams, which are illustrated in the module's interface. In addition, it is needed to keep this entire process hide from the user, as the logic behind the knowledge management it is something with a high complexity, as it is based on models and metamodels.

1.2 Objectives

The main objective of this project will establish the platform to manage and illustrate all information types, previously stored via the RSHIP model.

As illustrating all information types is a really difficult task to accomplish, the project just will take care of adding images to the different types of elements shown in the diagrams (as for example giving to electrical and electronic circuit´s elements representation with icons) and creating new layouts which will show new ways of illustrating the information stored on RSHIP model. In particular, it is intended to give the module the circuit´s diagram visual structure as improvement to its functionality.

With this new layout, it will take not too long to develop new layouts to represent any information type, as needed by the organization in the future, as this new layout will be easy to modify to implement new ones.

2. Estado del arte

Este capítulo es fundamental para entender las necesidades de este proyecto, ya que en él se explicará su marco tecnológico, explicando la plataforma en la que se asienta este proyecto.

Así se podrán aclarar los conceptos teóricos para poder entender el proyecto de la forma más clara posible.

A continuación se hablará de: gestión del conocimiento. Lenguaje RSHIP, la herramienta Knowledge Manager y las tecnologías usadas para su desarrollo.

2.1 Gestión del conocimiento

A lo largo de su existencia la humanidad ha tenido el deseo de gestionar su conocimiento, y la evolución de la sociedad, ha pasado de ser una sociedad de la **información** a una sociedad del **conocimiento.**

Así en las organizaciones, las personas y la gestión del conocimiento se han convertido en la ventaja competitiva, sustituyendo el paradigma anterior basado, por ejemplo, en la automatización.
Esto ha sido posible gracias a la irrupción de las TIC en el siglo XXI, que empiezan a tratar toda la información que contienen de tal manera que se convierte en conocimiento efectivo que puede llegar a ser un activo determinante para la competitividad de la empresa.

Se trata de averiguar que se sabe realmente, para apalancar el objetivo de negocio de la misma.

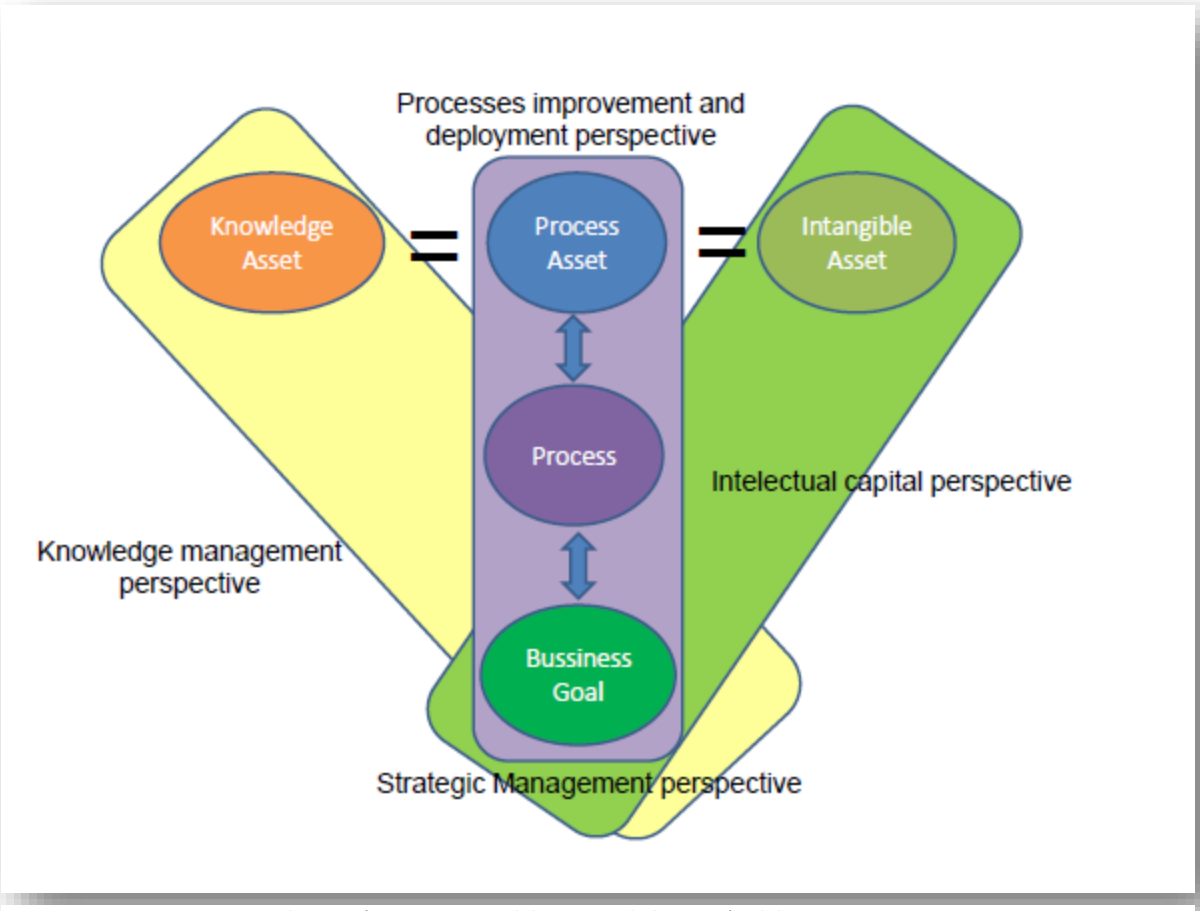

Ilustración 2: Esquema del proceso de la gestión del conocimiento

Desde un punto de vista más práctico, las herramientas en las que se basa la gestión del conocimiento son:

- *Herramientas para el almacenamiento y organización de la información.*
- *Herramientas de recuperación y filtrado de la información de manera eficiente.*
- *Herramientas de edición, personalización y mejora de la información.*
- *Herramientas de análisis.*

Gracias al uso combinado de estas herramientas se logra la transformación de la información de una organización, en conocimiento con gran valor para la misma, que puede ser reutilizado y mejorado continuamente.

Existen numerosas herramientas y plataformas usadas en la gestión del conocimiento, pero es necesario delimitar el marco operacional en el que se mueve este proyecto.

Así el Knowledge Manager, es una herramienta que permite a empresas con grandes procesos industriales y de desarrollo (clientes como Airbus por ejemplo), reducir el coste de la gestión de

la información, a pesar del proceso ingenieril y su consiguiente volumen de conocimiento a gestionar.

2.2 Knowledge Manager

Como se mencionaba anteriormente el Knowledge Manager es una herramienta que permite la gestión del conocimiento, gracias al uso combinado de varias herramientas para la gestión del conocimiento, facilitando la tarea de manera considerable a los usuarios.

Así el Knowledge Manager ofrece un potente gestor de ontologías con el que se permite definir y gestionar la semántica contenida en un proyecto.

De una manera más coloquial, el Knowledge Manager consigue gestionar el conocimiento de un proyecto para que este sea reusable por el usuario en otros tantos y dando valor por tanto a la organización.

Esto se consigue mediante la combinación de conocimiento y requisitos, obteniendo conocimiento a través de los requisitos especificados en la aplicación, así como extrayendo nuevos requisitos del nuevo conocimiento que adquiere la organización, o de la mejora del conocimiento existente.

Para llegar a comprender el Knowledge Manager correctamente, es necesario definir los siguientes conceptos en los que se basa el mismo.

2.2.1 Ontología

Una ontología se define como un medio para la facilitar la comunicación y el intercambio de información entre diferentes sistemas y entidades, partiendo siempre de un riguroso y exhaustivo esquema conceptual dentro de uno o varios dominios de datos.

El Knowledge Manager ofrece un potente gestor de ontologías con el que se permite definir las principales semánticas de un proyecto, y aplicar sobre el dominio o ámbito de conocimiento que se define en el proyecto un método con el fin de obtener una representación formal de los conceptos que contiene y de las relaciones que existen entre estos.

Como gestor de ontologías dentro del Knowledge Manager se usa un sistema denominado Knowledge Organization System, o Sistema de Organización del Conocimiento, y entre sus funcionalidades más importantes, se encuentra la de importar diferentes tipos de información de otras herramientas y de exportar la información en diferentes tipos de archivo para que sean accesibles por otra herramientas.

A continuación se describirán los componentes de esta herramienta, según las especificaciones recogidas en un breve cursillo con Juan Llorens acerca del sistema:

- **Gestión del dominio:** Encargado de gestionar todos los aspectos relacionados con el dominio, que a su vez se subdividen en diferentes módulos:
  - Terminología.
  - Tokenización.
  - Normalización.
  - Etiquetado.
  - Ontología ligera.
  - Reglas de inferencia o patrones.

- **Subdominios**: En este módulo se encuentran todos los aspectos relacionados con la gestión de sub-dominios, grupos de reglas de inferencia y el tesauro del dominio.
- **Indización y motor de búsqueda**: Encargado del guardado y clasificación de la información y de la recuperación y filtrado de la misma.
- **Documentos**: En este módulo se reconocen todos los diferentes artefactos que pueden ser tratados en la ontología, es decir, se gestionan los distintos objetos de datos recogidos por la herramienta en la ontología.
- **Configuración**: Se encarga de todos los aspectos relacionados con la configuración de la herramienta, estableciendo el usuario parámetros como por ejemplo filtros de búsqueda.

Sin embargo, para llegar a entender el Knowledge Manager, es necesario entender como la herramienta es capaz de entender tantos tipos de información, así como de gestionarla.

Todo esto es gracias al modelo de representación que se define mediante el lenguaje conocido como RSHIP (anteriormente RSHP).

2.3 RSHIP

RSHIP como se ha mencionado anteriormente, es el modelo de representación de la información basado en relaciones (en inglés **Relationships**, del que se define este acrónimo).
Este modelo intenta cubrir la necesidad de representar cualquier tipo de información usando un esquema de representación común, intentando generalizar el manejo de cualquier tipo de información.

Simplificando la introducción anterior, este modelo se basa en la relación entre conceptos, siendo la relación la que tiene el papel principal de este modelo.
Entre los elementos que se relacionan se pueden distinguir dos tipos fundamentales, los términos (elemento simple) y los artefactos (elementos con contenido), y estos constituyen las unidades mínimas de la información.
Así se distingues tres tipos de elementos en este modelo:

- **Términos**: elementos básicos que representan el vocabulario del dominio.
- **Artefacto**: en RSHP son contenedores de información, actúan como términos pero con mayor información en su interior. La información almacenada en esto elementos llega desde un simple término a incluso otro modelo completo RSHIP.
- **Relación**: el elemento esencial del modelo RSHIP, ya que es el elemento que dota de sentido al modelo. Su misión es conectar los dos elementos anteriormente mencionados, describiendo la información almacenada en el modelo.

2.4 Sistema universal de edición gráfica

Este módulo se encarga de representar gráficamente el conocimiento, el cual es operado mediante el Knowledge Manager.

Siendo la base de este proyecto de fin de grado, este módulo desarrollado por mis compañeros el año pasado permite representar la información contenida en los modelos RSHIP.

Gracias a este módulo un usuario puede ver rápidamente una representación gráfica de la información contenida en el Knowledge Manager mediante librerías JavaScript y HTML5, que toman los elementos, términos y artefactos, como nodos a representar con relaciones entre ellos.

La decisión de usar JavaScript para las representaciones se debe a que este lenguaje es capaz de mover con fluidez grandes cantidades de elementos gráficos, permitiendo además en un futuro llevar este módulo a navegadores web.

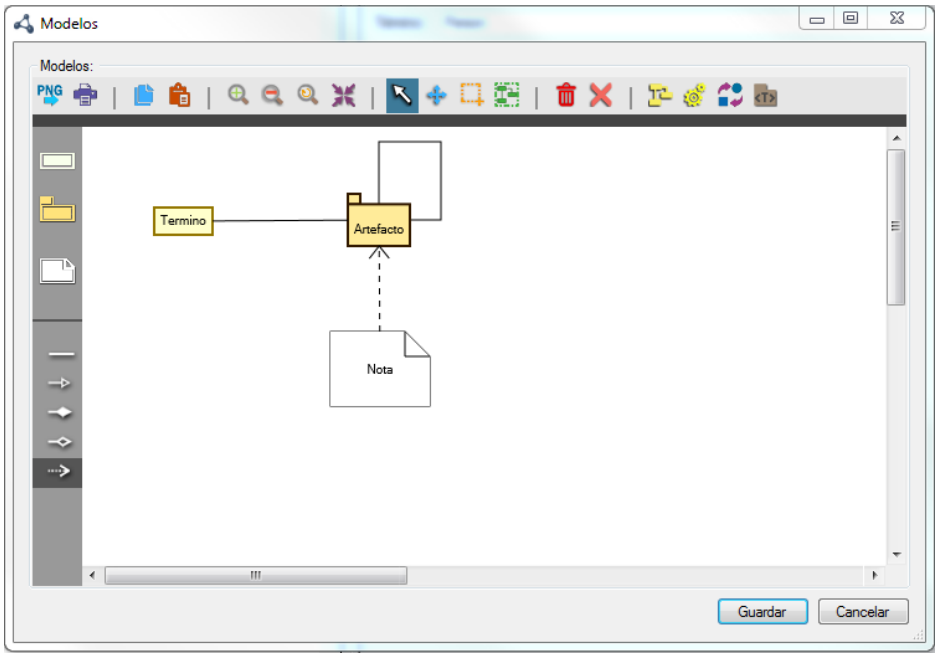

Ilustración 3: Pantalla del módulo de representación gráfica con los elementos principales representados

2.4.1 Librerías gráficas

La definición de una librería en informática consiste en un conjunto de funcionalidades implementadas en un cierto lenguaje de programación, y que proveen de una interfaz bien definida para que un programador haga uso de todas estas funcionalidades abstrayéndose de la programación de la librería.

Siguiendo esta definición, una librería permite proporcionar una interfaz que pueda ser usada por varios programas que no tienen conexión unos con otros.

Así mis compañeros del año pasado investigaron las diferentes librerías que existían en el mundo JavaScript para poder representar y gestionar los elementos gráficos de este módulo, y eligieron la librería **Lassalle AddFlow**, debido a la amplia funcionalidad que provee así como el relativo bajo coste de la licencia. Más adelante se detallarán en profundidad las características de esta librería.

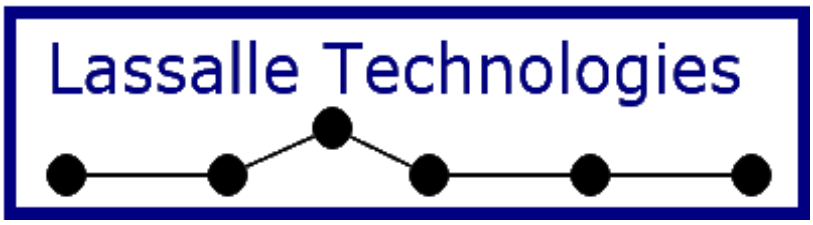

Ilustración 4: Logotipo de la empresa detrás de la librería JavaScript Lassalle AddFlow

Incluso, incorpora numerosas funcionalidades las cuales pueden ser aplicadas en el modelo, como así recogieron en la documentación mis compañeros del año pasado:

- Crear nodos.
- Crear relaciones.
- Modificar nodos y relaciones existentes.
- Exportar el gráfico en distintos tipos de formato (jpg, jpeg y gif).
- Imprimir un gráfico.
- Copiar un gráfico
- Pegar un gráfico.
- Aumentar, disminuir o, autoajustar zoom.
- Seleccionar varios elementos.
- Seleccionar todos los elementos.
- Eliminar nodos o relaciones.
- Eliminar todo el gráfico.
- Realizar esquematización del gráfico.
- Aumentar o disminuir el grado de información que se puede ver de forma gráfica.
- Cambiar el tipo de gráfico (Diagrama de clases UML, Diagrama de secuencia, etc…).
- Transformar diagrama. Es aquí donde se centra nuestro proyecto, el cual se encarga de realizar todas las operaciones necesarias para poder visualizar el gráfico según el tipo.

2.4.2 Objetivo de la mejora

El objetivo de este proyecto consiste en modificar la librería **Lassalle,** así como la aplicación en si para poder asentar la base para dibujar cualquier tipo de representación, llegando incluso a poder configurar la filosofía de dibujo de la misma según parámetros que introduzca el cliente en el módulo, pudiéndose elegir por ejemplo, si se quiere representar la información como un circuito eléctrico, un diagrama UML, o un nuevo tipo de diagrama que defina el usuario.

Como la complejidad de este objetivo es muy grande, y es un objetivo que se busca a largo plazo, este proyecto estará limitado a añadir representaciones en forma de iconos a los nodos que se dibujan en el gráfico, así como añadir una nueva filosofía de dibujo (circuito eléctrico)

dando las claves para así poder desarrollar nuevas filosofías de dibujo que no se puedan representar a través de la propia librería JavaScript.

Cabe destacar que una filosofía de dibujo se refiere a la disposición que tendrán los nodos en el módulo de representación, mostrando a estos, y a sus relaciones, colocados de tal manera que representen diagramas UML, circuitos eléctricos, diagramas de secuencia…

3. Estudio de viabilidad

En este apartado de estudio de viabilidad del sistema trata de asegurar que el proyecto propuesto es factible desde el punto de vista técnico, económico y operativo, y cumple con las necesidades del cliente, en este caso Juan Llorens.

Así se describirán el sistema y las alternativas necesarias de cara al desarrollo del proyecto, contrastándolas para así ofrecer al cliente aquella que presenta una mejor solución a sus necesidades.

A continuación se detallará la situación actual del sistema, con el objetivo de llevar a cabo un análisis de los sistemas realizados en la tarea anterior, identificando problemas existentes, deficiencias y mejoras. Estás ultimas se tendrán en cuenta para la definición de requisitos.

3.1 Situación actual del sistema

Como se ha descrito anteriormente, en este punto se realizará un diagnóstico de la situación actual del proyecto, con el objetivo de localizar mejoras y soluciones a errores encontrados en la tarea anterior.

En el proyecto anterior se pretendía desarrollar un módulo de representación gráfica del modelo del cliente RSHIP como se ha detallado en el apartado introductorio del estado del arte.

Este proyecto se podría catalogar perfectamente como un proyecto desarrollado en distintas etapas, siendo la primera etapa la tarea anterior a este proyecto actual.
El objetivo fundamental de este módulo es llegar a representar cualquier tipo de información almacenada en el Knowledge Manager, buscando representar la universalidad de la información al usuario de manera gráfica.

Partiendo de este objetivo, se realizará un análisis del estado en el que se encuentra el módulo actualmente, detectando que necesidades están cubiertas para esta etapa, y cuáles no.

En un principio esté módulo cumple con las necesidades del cliente de poder representar cualquier tipo de información en lenguaje nativo, es decir, representando la información siguiendo el modelo definido por RSHIP. Además se puede modificar la información representada y añadir nueva información, con opciones de configuración para el usuario como añadir nuevos elementos como son las relaciones, los términos o los artefactos.
Con respecto a este aspecto, el proyecto no presenta deficiencias, ya que cumple con todas las funcionalidades de representación RSHIP definida por el cliente.

Sin embargo, revisando las necesidades de esta etapa del módulo, se han detectado diferentes deficiencias que este proyecto pretende solucionar.

La primera consiste en que aunque se consiga representar cualquier tipo de información de manera nativa (RSHIP), el sistema no es capaz de mostrar está información en cualquier filosofía de dibujo que defina el usuario. De manera más coloquial, el sistema está limitado a representar exclusivamente información siguiendo la filosofía de dibujo UML, diagramas de secuencia o de manera nativa.

Así por ejemplo si un cliente quiere representar un circuito eléctrico, podrá representar toda esta información de manera nativa, definiéndola a través del Knowledge Manager, o incluso arrastrando elementos nuevos al módulo de representación gráfica, pero no podrá mostrar estos elementos definidos en forma de un esquema eléctrico. Véase que sucede lo mismo con cualquier representación que no sea UML, diagrama de secuencia o el tipo de representación nativa.

Desde un punto de vista gráfico, este es un gran problema para el usuario, ya que dificulta al mismo de abstraerse de la información que se le está representando en el módulo.

Así que una de las principales mejoras de este proyecto será poder definir más formas de representar está información una vez definida, mediante la función que ya proporciona el módulo, que es la opción de transformación del diagrama nativo en la forma de dibujo que requiera el usuario.

Se han encontrado además algunos fallos más con respecto a funciones como el borrado de todos los elementos representados por el módulo, así como algunos problemas de edición de los mismos elementos.

Otro aspecto que no ha cubierto la tarea anterior, es un problema de estética del proyecto, y consiste en que el usuario no puede añadir representaciones a los elementos de información que define.

Añadir esta funcionalidad permite además de dar mucha más información de un elemento a través de su representación, ya que permite identificarlo mucho más fácilmente, añadir un

componente visual que es muy atractivo para el usuario, dejando un gráfico estéticamente más bonito y claro para sus necesidades.

Siendo estas las principales mejoras detectadas en el proyecto anterior, y tras presentar estas posibles mejoras al cliente, se ha acordado que en esta etapa del módulo, se centre en mejorar las funciones existentes mediante la adición de imágenes (representaciones) de los elementos que se muestran en el gráfico, permitir definir nuevas representaciones mediante la configuración de parámetros por parte del cliente, y por último añadir una nueva filosofía de dibujo para asentar la base para poder añadir en un futuro todas las nuevas filosofías de dibujo que requiera el cliente según sus necesidades. Está filosofía de dibujo será la de representar un circuito eléctrico.

Esto se debe a que un gran número de clientes del Knowledge Manager lo han requerido y el cliente ha considerado que es una mejora altamente valorable para el sistema, delimitando así por parte del cliente la extensión de las mejoras sobre el proyecto desarrollado anteriormente.

Partiendo de estas necesidades, se establecerá la base para la especificación de requisitos de usuario.

3.2 Especificación de requisitos del sistema

En este apartado se realizará un exhaustivo estudio sobre los requisitos del sistema teniendo en cuenta en un principio los que define el usuario.
El objetivo de estos requisitos es garantizar que sean correctos y atiendan a las necesidades del cliente, por lo que es muy importante que los requisitos sean bien definidos por su parte ya que tiene una figura muy importante en el desarrollo de todo el proyecto.

Para la identificación de los requisitos de usuario distinguiremos tres tipos de requisitos:

- **Requisitos de capacidad**: Son aquellos que resuelven un problema que pueda ser planteado por el usuario consiguiendo un resultado satisfactorio.
- **Requisitos de restricción**: Recogen las condiciones que imponen los usuarios para poder llegar al objetivo que han planteado o al método de resolución de un problema.
- **Requisitos inversos**: Son requisitos que indican casos por los que el módulo desarrollado no debe pasar en ningún caso, se definirán con el consenso con el cliente.

Hay más tipos de requisitos del sistema que se especificarán en el apartado de análisis ya que en este punto sólo se detallaran los relacionados con los requisitos definidos por el usuario con el fin de comprender con más facilidad la viabilidad de la propuesta. Todos los requisitos recogidos en este punto se basan en los mismos aprobados entre los encargados de la anterior etapa del proyecto y el cliente, con la adición de los nuevos que se han especificado en esta etapa. Se considera importante incluir los requisitos aprobados en la etapa anterior junto a los

nuevos para poder entender mejor las mejoras aplicadas al sistema así como las modificaciones que se aplicarán al módulo.

Se ha considerado interesante indicar cuales son los nuevos requisitos de usuario para diferenciarlos de los recogidos en la etapa anterior, aunque en la especificación de requisitos de sistema que se recoge en este documento, se muestra las dependencias de estos requisitos de sistema con los requisitos de usuario de la etapa anterior, junto con los que se han recogido en esta nueva etapa. Con ello se consigue demostrar que se mantienen siempre las necesidades del usuario en cualquier etapa de desarrollo de este proyecto.

Para cada requisito se creará una tabla con todos los campos que aportan información del mismo. Para estos campos se ha decidido basarse en Métrica 3, que define un estándar para la creación de documentos de gestión de proyectos y es la oficialmente usada en la Universidad Carlos III.

Los campos que tendrán cada tabla de requisitos son los siguientes:

- **Identificador**: permitirá identificar a cada requisito para diferenciarlos, como por ejemplo RU-XYY donde cada campo significa:
  - RU: Requisito de Usuario
  - YY: Numeración que identifica al requisito.
  - X: Campo que indica el tipo de requisito dentro de los requisitos de usuario. Por ejemplo: RU-C01. Indica el requisito de usuario de capacidad número 1.
  - Según el tipo tendremos por lo tanto:
    - RU-CYY: Requisitos de usuario de capacidad.
    - RU-RYY: Requisitos de usuario de restricción.
    - RU-IYY: Requisitos de usuario inversos.

- **Fuente**: Recoge quien ha identificado al requisito. En este caso, la fuente sólo podrá ser el cliente (Juan Llorens) o el desarrollador del proyecto (Alberto Guzmán Aroca).

- **Título**: Nombre al que es referido el requisitos. Ej.: Nodos con imagen sin bordes.

- **Necesidad**: Esta campo permitirá medir la necesidad que tiene el cliente para que se cumpla este requisito. Se distinguirán tres niveles que son: Esencial, deseable y opcional. Siendo los esenciales como requisitos que no deben modificarse a lo largo del proyecto.

- **Prioridad**: Cada requisito tendrá un campo de prioridad, ya que según si es más alta o más baja el desarrollador podrá planificar de manera más sencilla las tareas de este proyecto. Existirán tres niveles: Alta, Media y Baja.

- **Verificabilidad**: Cada requisito deberá contener este campo para indicar si se ha integrado correctamente en el diseño del sistema.

A continuación se mostrará una plantilla de cómo se recogerá un requisito en este documento, y será la misma para cada requisito definido en este documento.

ID: RU-XYY			
Fecha	Dd/mm/aaaa	**Título**	Título del requisito
Necesidad	Esencial	Deseable	Opcional
Prioridad	Alta	Media	Baja
Verificabilidad	Alta	Media	Baja
Descripción	Descripción breve del requisito		
Fuente	El cliente o el desarrollador		

Tabla 1: Ejemplo de tabla para requisitos

3.2.1 Requisitos de capacidad

ID: RU-C01			
Fecha	23/02/2015	**Título**	Visualización gráfica de información de cualquier tipo
Necesidad	Esencial	Deseable	Opcional
Prioridad	Alta	Media	Baja
Verificabilidad	Alta	Media	Baja
Descripción	Se podrá visualizar en el módulo cualquier tipo de información que se encuentre representada mediante el modelo RSHIP.		
Fuente	The Reuse Company		

Tabla 2: RU-C01: Visualización gráfica de información de cualquier tipo.

ID: RU-C02			
Fecha	23/02/2015	**Título**	Edición de la información gráfica
Necesidad	Esencial	Deseable	Opcional
Prioridad	Alta	Media	Baja
Verificabilidad	Alta	Media	Baja
Descripción	El usuario podrá editar la información gráfica cuando se represente mediante un modelo de representación universal		
Fuente	The Reuse Company		

Tabla 3: RU-C02: Edición de la información gráfica

ID: RU-C03				
Fecha	23/02/2015		**Título**	Creación de cualquier tipo de información gráfica
Necesidad	Esencial	Deseable		Opcional
Prioridad	Alta	Media		Baja
Verificabilidad	Alta	Media		Baja
Descripción	El usuario podrá crear cualquier tipo de información gráfica cuando se represente mediante un modelo de representación universal			
Fuente	The Reuse Company			

Tabla 4: RU-C03: Creación de cualquier tipo de información gráfica

ID: RU-C04				
Fecha	23/02/2015		**Título**	Gestión de información gráfica
Necesidad	Esencial	Deseable		Opcional
Prioridad	Alta	Media		Baja
Verificabilidad	Alta	Media		Baja
Descripción	El usuario podrá gestionar la información gráfica mediante una interfaz de usuario.			
Fuente	The Reuse Company			

Tabla 5: RU-C04: Gestión de información gráfica

ID: RU-C05				
Fecha	23/02/2015		**Título**	Almacenado de la información
Necesidad	Esencial	Deseable		Opcional
Prioridad	Alta	Media		Baja
Verificabilidad	Alta	Media		Baja
Descripción	El usuario podrá persistir la información gráfica.			
Fuente	The Reuse Company			

Tabla 6: RU-C05: Almacenado de la información

ID: RU-C06			
Fecha	23/02/2015	**Título**	Visualización de información transformada
Necesidad	Esencial	Deseable	Opcional
Prioridad	Alta	Media	Baja
Verificabilidad	Alta	Media	Baja
Descripción	El usuario podrá visualizar la información gráfica en su modo nativo para tipos especificados con anterioridad		
Fuente	The Reuse Company		

Tabla 7: RU-C06: Visualización de información transformada

ID: RU-C07			
Fecha	23/02/2015	**Título**	Exportado de la información gráfica
Necesidad	Esencial	Deseable	Opcional
Prioridad	Alta	Media	Baja
Verificabilidad	Alta	Media	Baja
Descripción	El usuario podrá exportar información gráfica		
Fuente	The Reuse Company		

Tabla 8: RU-C07: Exportado de la información gráfica

ID: RU-C08			
Fecha	23/02/2015	**Título**	Localización del sistema
Necesidad	Esencial	Deseable	Opcional
Prioridad	Alta	Media	Baja
Verificabilidad	Alta	Media	Baja
Descripción	El sistema a desarrollar debe saber en qué localización geográfica se está ejecutando		
Fuente	The Reuse Company		

Tabla 9: RU-C08: Localización del Sistema

ID: RU-C09			
Fecha	23/02/2015	**Título**	Hacer y deshacer
Necesidad	Esencial	Deseable	Opcional
Prioridad	Alta	Media	Baja
Verificabilidad	Alta	Media	Baja
Descripción	El usuario podrá deshacer y rehacer las acciones efectuadas sobre la información gráfica		
Fuente	The Reuse Company		

Tabla 10: RU-C09: Hacer y deshacer

- **Nuevos requisitos de capacidad recogidos en esta etapa de desarrollo:**

ID: RU-C10			
Fecha	23/02/2015	**Título**	Layout
Necesidad	Esencial	Deseable	Opcional
Prioridad	Alta	Media	Baja
Verificabilidad	Alta	Media	Baja
Descripción	El sistema podrá realizar una recolocación lógica de la información gráfica, según la filosofía de dibujo tanto en la vista nativa como la transformada		
Fuente	The Reuse Company		

Tabla 11: RU-C10: Layout

ID: RU-C11			
Fecha	23/02/2015	**Título**	Seleccionar tipo de Layout
Necesidad	Esencial	Deseable	Opcional
Prioridad	Alta	Media	Baja
Verificabilidad	Alta	Media	Baja
Descripción	El usuario podrá elegir el layout, entre las opciones dadas, de la información gráfica con la que trabaje en el módulo.		
Fuente	The Reuse Company		

Tabla 12: RU-C11: Seleccionar tipo de Layout

ID: RU-C12			
Fecha	23/02/2015	**Título**	Layout de circuito eléctrico
Necesidad	Esencial	Deseable	Opcional
Prioridad	Alta	Media	Baja
Verificabilidad	Alta	Media	Baja
Descripción	El sistema podrá realizar una recolocación lógica de la información gráfica en forma de circuito eléctrico.		
Fuente	The Reuse Company		

Tabla 13: RU-C11: Layout de circuito eléctrico

ID: RU-C13			
Fecha	23/02/2015	**Título**	Cargar iconos en los nodos
Necesidad	Esencial	Deseable	Opcional
Prioridad	Alta	Media	Baja
Verificabilidad	Alta	Media	Baja
Descripción	El sistema podrá definir nodos que tengan representación gráfica (iconos) o layouts.		
Fuente	The Reuse Company		

Tabla 14: RU-C12: Cargar iconos en los nodos

ID: RU-C14			
Fecha	23/02/2015	**Título**	Cambios en el diagrama
Necesidad	Esencial	Deseable	Opcional
Prioridad	Alta	Media	Baja
Verificabilidad	Alta	Media	Baja
Descripción	El sistema mantendrá cualquier cambio en el diagrama entre modo nativo y el modo transformado.		
Fuente	The Reuse Company		

Tabla 15: RU-C14: Cambios en el diagrama

3.2.2 Requisitos de restricción

ID: RU-R01			
Fecha	23/02/2015	**Título**	Restricción de entorno operativo
Necesidad	Esencial	Deseable	Opcional
Prioridad	Alta	Media	Baja
Verificabilidad	Alta	Media	Baja
Descripción	El sistema deberá poder ejecutarse en un entorno Win32		
Fuente	The Reuse Company		

Tabla 16: RU-R01: Restricción de entorno operativo

- **Nuevo requisito de restricción**

ID: RU-R02			
Fecha	23/02/2015	**Título**	Tamaño máximo de las imágenes
Necesidad	Esencial	Deseable	Opcional
Prioridad	Alta	Media	Baja
Verificabilidad	Alta	Media	Baja
Descripción	El sistema deberá limitará el tamaño de las imágenes que pueda añadir el usuario en los nodos contenidos en la información gráfica.		
Fuente	The Reuse Company		

Tabla 17: RU-R02: Tamaño máximo de las imágenes

3.2.3 Requisitos inversos

ID: RU-I01			
Fecha	23/02/2015	**Título**	Funcionamiento como subsistema
Necesidad	Esencial	Deseable	Opcional
Prioridad	Alta	Media	Baja
Verificabilidad	Alta	Media	Baja
Descripción	El sistema no deberá funcionar fuera del Knowledge Manager, siendo un sistema embebido del mismo.		
Fuente	The Reuse Company		

Tabla 18: RU-I01: Funcionamiento como subsistema

ID: RU-I02			
Fecha	23/02/2015	**Título**	Imágenes no almacenadas como recurso local
Necesidad	Esencial	Deseable	Opcional
Prioridad	Alta	Media	Baja
Verificabilidad	Alta	Media	Baja
Descripción	El sistema no deberá almacenar como recurso local las imágenes que el usuario pueda añadir a los nodos del diagrama		
Fuente	The Reuse Company		

Tabla 19: RU-I02: Imágenes no almacenadas como recurso local

3.3 Estudio de alternativas tecnológicas para el proyecto

En este punto se realizará un análisis de las posibles alternativas a este sistema, aunque el cliente ha definido que se siga claramente con la línea elegida en la etapa anterior. Por lo tanto este documento recoge la alternativa que fue seleccionada en la etapa anterior de desarrollo del módulo.

Toda la selección de alternativas de la etapa anterior viene recogida en la documentación entregada por mis compañeros el año pasado, por lo que si se quiere saber cómo se llegó a esta alternativa lo mejor es consultar sus memorias que vienen especificadas en la bibliografía de este documento.

A continuación se detallarán las tecnologías usadas durante el desarrollo de este módulo, detallando tanto las diferentes **herramientas tecnológicas para el desarrollo**, así como también las que se usarán para la **gestión del proyecto**.

Cabe destacar que ante la imposibilidad de seleccionar otras alternativas, la dificultad de desarrollo de este proyecto se ha visto incrementada considerablemente, ya que los conocimientos aplicados en este módulo podrían haberse desarrollado en otras herramientas y otros entornos de programación en los que se cuenta con más experiencia. Además, el esfuerzo en el aprendizaje de esta alternativa ha sido muy alto, disminuyendo el esfuerzo que se debería haber dedicado a otras tareas como la investigación de los diferentes algoritmos necesarios para cumplir con las expectativas del cliente.

3.3.1 Tecnologías necesarias para el desarrollo

En este apartado se analizarán las diferentes tecnologías que intervienen en el desarrollo de este módulo, teniendo en cuenta las ventajas y desventajas con las que cuenta cada una.

3.3.1.1 Sistema operativo

El sistema operativo es el conjunto de programas que gestiona los recursos de hardware y provee de servicios a los programas de aplicación.

Para el análisis del sistema operativo se van a seguir los siguientes criterios:

- **Licencia:** Según el tipo de contrato entre el usuario final del sistema operativo y los propietarios del mismo, está puede ser Open Source o propietaria.
- **Coste:** Precio de adquirir la licencia, se evaluará su precio en euros.
- **Experiencia**: Experiencia que el desarrollador tiene sobre el sistema operativo, podrá ser alta, media o baja.

El sistema operativo seleccionado para el desarrollo de este módulo será el siguiente:

- **Windows 8.1:** Última versión del sistema operativo desarrollado por la empresa Microsoft.

A continuación se mostrará una tabla con los diferentes criterios mencionados anteriormente para el sistema operativo.

SO	Licencia	Coste	Experiencia
Windows 8.1	Propietaria	119 € (Real 0€)	Alta

Tabla 20: Tabla con especificaciones de las alternativas para el sistema operativo

Como se aprecia en la tabla, se ha seleccionado la solución basada en Windows 8.1. Sobre este sistema se cuenta con una gran experiencia, sin contar que aunque la licencia sea la muy cara (119 €), se puede conseguir gratuitamente gracias a un convenio de Microsoft con la Universidad Carlos III de Madrid (el programa MSDN).

3.3.1.2 Lenguaje de programación

Un lenguaje de programación es un lenguaje formal diseñado para expresar procesos que pueden ser llevados a cabo, por ejemplo, por este mismo módulo mediante la funcionalidad que ofrece en un terminal cualquiera (cumpliendo con los requisitos de Sistema Operativo especificados en el punto anterior).

En este apartado describiremos las diferentes opciones que se han barajado para la programación de las nuevas funcionalidades que debe recoger la aplicación y que están especificadas en los requisitos de usuario anteriormente definidos en el punto 3.2, teniendo en cuenta que se deben mantener estas funcionalidades que ya existen en el módulo sin que se vean afectadas. Así, y considerando estas necesidades, se elegirá el mismo lenguaje de programación usado en la etapa anterior del proyecto, que será **HTML5** combinado con **JavaScript**.

Además se usarán también como apoyo:

- **CSS**: Este lenguaje es usado para definir como se deben mostrar los elementos HTML de una página, es decir, define la presentación de un documento estructurado HTML.
- **JQuery**: Siendo una de las librerías JavaScript más extensamente usadas, esta permite simplificar la interacción con los documentos HTML, manipular el árbol DOM, manejar eventos e incluso desarrollar animaciones, entre otras muchísimas funciones.
- **PowerTip**: Una pequeña librería JavaScript que se eligió en la etapa anterior del proyecto, y que permite gestionar mensajes de ayuda asociados a elementos HTML, en esta etapa se seguirá usando para crear los tooltips del módulo.

Por último hay que mencionar que en este módulo se usa sólo HTML5 y JavaScript para el apartado de visualización de los elementos gráficos. Así el controlador, que será la parte que administre la comunicación entre datos persistidos por el usuario y que se mostrarán en la vista (básicamente el HTML y el JavaScript), para que el mismo usuario pueda visualizarlos y gestionarlos mediante la interfaz, está programado enteramente en C# usando .NET Framework.

No se detallarán más alternativas para la programación de la parte controladora, ya que es un requisito establecido en la fase de análisis y consensuado con el cliente. Esté está recogido en el apartado cuatro de este documento.

Ilustración 5: Los tres principales lenguajes de programación usados para el módulo de representación gráfica

3.3.1.3 Entorno de programación

El entorno de programación es la plataforma software o *framework* en el que se programa y se compila un determinado lenguaje o varios.

Para evaluar los diferentes entornos de programación que surgen como alternativas para este proyecto, se usará una tabla con los mismos campos que se detallaron en el apartado 3.3.1.1 para especificar las alternativas al sistema operativo en el que se asentará el módulo.

En este caso el IDE impuesto por el cliente será Microsoft Visual Studio 2010 en su versión Ultimate, cuyas características se detallan a continuación:

IDE	Licencia	Coste	Experiencia	¿Compatible con JavaScript?
Microsoft Visual Studio 2010 Ultimate	Propietaria	729 € (Real 0 €)	Media	Sí

Tabla 21: Tabla con especificaciones de los IDEs

Debido a los requisitos del cliente, la alternativa seleccionada será Visual Studio, ya que toda la aplicación del Knowledge Manager ha sido desarrollada en este IDE, por lo que la manera más sencilla de integrar el módulo de representación gráfica mejorado consiste en seguir con la línea del Knowledge Manager.

Aunque el coste del Visual Studio 2010 parece muy alto, gracias al acuerdo entre la Universidad Carlos III de Madrid y Microsoft, la licencia de este software es gratuita para los alumnos, porque la solución más idónea sería usar Visual Studio como entorno de programación.

3.3.2 Gestión del proyecto

A continuación se detallarán las diferentes alternativas que se han seleccionado para la gestión de este proyecto, y que es un punto crucial para la consecución del mismo.

Para ello se detallarán las herramientas que se usarán para las diferentes tareas necesarias en la gestión de proyecto.

3.3.2.1 Documentación

Para recoger toda la documentación que se ha redactado en este proyecto se han usado las aplicaciones que provee el paquete ofimático de Microsoft Office. Esto se debe a la extensión de uso de estas herramientas, que junto con la gran experiencia que se adquiere en su uso durante la carrera hace que sean las aplicaciones ideales para documentar este proyecto.

Las aplicaciones de Microsoft Office utilizados son:

- Microsoft Word: Es un procesador de textos.
- Microsoft Excel: Permite la creación y edición de tablas y hojas de cálculos, así como gráficos.
- Microsoft PowerPoint: Gestor de contenido multimedia usado para la presentación de este proyecto.

Además de esta suite ofimática se usará también:

- Microsoft Project: Herramienta para la gestión y planificación de las diferentes tareas asignadas a este proyecto.

3.3.2.2 Diagramas

Para la gestión de diagrama se ha decidido usar la herramienta Microsoft Visio, que forma parta de la suite de Microsoft y es un potente gestor de diagramas, proporcionando medios para poder representar casi cualquier tipo de gráfico.

3.3.3 Gestión de versiones

Como se define en la página de Git (Herramienta de control de versiones):

"El control de versiones es un sistema que registra los cambios realizados sobre un archivo o conjunto de archivos a lo largo del tiempo, de modo que puedas recuperar versiones específicas más adelante."

Esto permite que en caso de que ocurra algún problema, se puede retornar a una versión anterior del mismo documento o archivo para así no perder todo el trabajo desarrollado sobre él hasta el momento.
También la gestión de versiones permite trabajar a un número grande de personas sobre el mismo proyecto evitando los problemas de conflictos (gracias a las copias de seguridad), así como también permite actualizar a todos los componentes del equipo los últimos cambios realizados en los documentos y archivos. Aunque no tenga mucha utilidad para el desarrollo de este proyecto al ser un proyecto individual, permite tener en una nube privada todo el proyecto por lo que es mucho más fácil acceder a él desde cualquier terminal.

Para la gestión de versiones se ha decidido usar la herramienta "Tortoise SVN", que es un cliente de Subversion que instalado en un servidor que funcione 24 horas al día permite crear la nube que se ha definido en el párrafo anterior, permitiendo acceder a todos los archivos de código, así como a todos los documentos en cualquier momento.

3.3.4 Gestión de pruebas

Es fundamental asegurar que el desarrollo de este módulo cuente con un grado mínimo de calidad, comprobando que el sistema cumpla con todas las funcionalidades acordadas entre el desarrollador y el cliente. Para ello se usará un plan de pruebas definiendo un conjunto de pruebas unitarias que recojan todas las funcionalidades posibles del sistema.
Se ha utilizado, siguiendo las líneas del año pasado "Nunit" como plataforma para la gestión de las pruebas unitarias, siendo opensource y con soporte para Microsoft .NET.

3.3.5 Gestión de tareas

Para llegar a hacer una correcta planificación de las tareas, y mantener un proyecto organizado desde el principio al final, controlando el esfuerzo que se emplea en cada tarea, así como el esfuerzo estimado para finalizarse, es interesante el uso de alguna herramienta para la gestión de tareas. La herramienta elegida es "Trello", que aunque no es un gestor de tareas al uso, proporciona unas herramientas para gestión de tareas asignadas a equipos de calidad.
Esta herramienta que se basa en listas, y permite añadir a los elementos de las listas cualquier cosa, desde más listas hasta imágenes. Estas funcionalidades junto a que es una aplicación web, hacen de esta herramienta un pilar fundamental para la gestión de este proyecto.
Además, al final del proyecto esta herramienta se convierte en un registro de todo lo que se ha ido haciendo.

Ilustración 6: Logo del gestor de tareas Trello

3.4 Librerías gráficas

En este punto se detallarán las características principales de la librería gráfica seleccionada en la etapa anterior. Como en herramientas detalladas anteriormente, su uso viene impuesto también por el cliente.

Cabe destacar que la necesidad de usar una librería gráfica en concreto, que ya viene modificada de la etapa anterior, implica un esfuerzo enorme para llegar a comprender como está

implementada, las posibilidades que tiene, y las diferentes modificaciones que se realizaron sobre la misma.

El aprendizaje de esta librería ha sido una tarea ardua y complicada, ya que la experiencia que se cuenta sobre implementaciones JavaScript tan complejas es muy pequeña en comparación con otras posibles soluciones a las necesidades con las que cuenta el cliente. Cabe destacar también lo deficiente que son los comentarios del código de las funciones de la librería y la carencia de un depurador JavaScript en condiciones debido a la necesidad de usar como IDE Visual Studio 2010.

Si se hubiera usado una versión más reciente del mismo, por ejemplo, Visual Studio 2013, la tarea de aprendizaje hubiera requerido muchísimo menos esfuerzo, ya que esta herramienta cuenta con soporte mucho mejor para tecnologías web como es JavaScript.

3.4.1 Lasalle Workflow

Lasalle Workflow es la librería que al final se seleccionó en la etapa anterior, en su versión para HTML5.

Esta librería implementada en JavaScript se basa en el uso de Canvas, siendo este un elemento HTML5 que se usa para dibujar gráficos vía scripting (es decir, usando JavaScript).

El uso de Canvas implica ciertos inconvenientes, ya que el código implementado para realizar los gráficos es muy poco reutilizable, por lo que el esfuerzo en programación es mucho mayor.

Aún con esta desventaja, Lasalle está enteramente programada en JavaScript y JQuery, dos plataformas cuyo uso está muy extendido y que cuenta con un gran número de documentación y de ejemplos en la red.

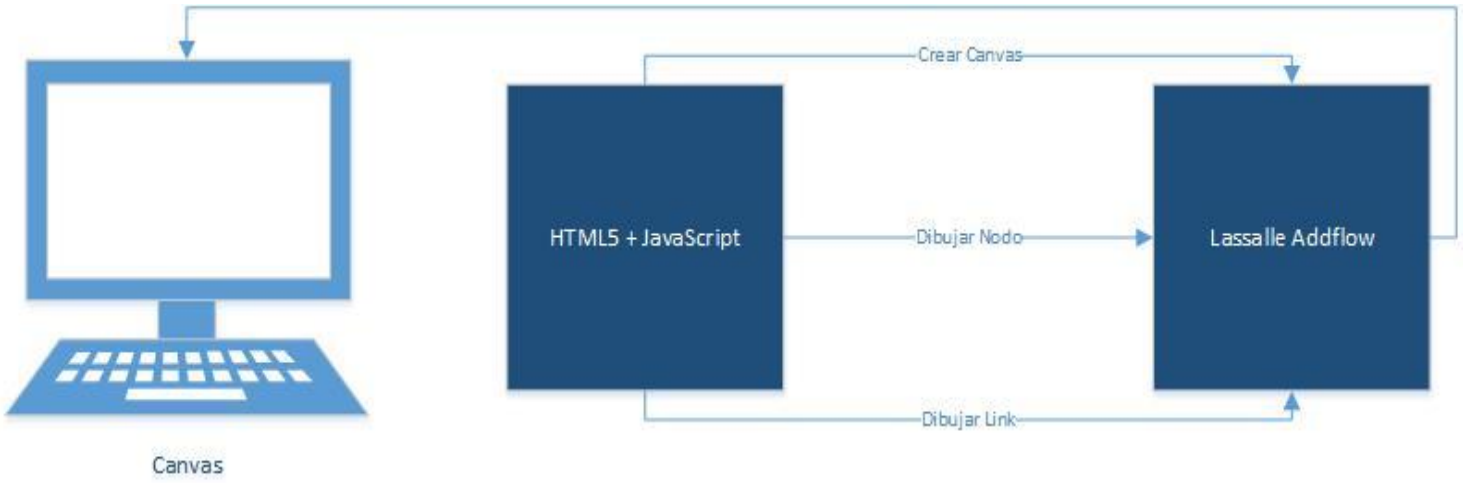

Ilustración 7: Funcionamiento y arquitectura para Lassalle

Como se muestra en el esquema de arriba, básicamente para que Lasalle funcione es necesario crear una página que servirá como capa de representación y donde se cargaran los script.

Una vez con la página creada se llama a la librería para que cargue el objeto que tenemos, permitiendo así añadir nodos y relaciones que la librería gestionará.

3.5 Representaciones para los nodos

Uno de los requisitos que el cliente estableció para este proyecto era que se diera la posibilidad al usuario de añadir representaciones en forma de iconos o imágenes a los nodos que conforman un diagrama.

Teniendo en cuenta esta necesidad, y sabiendo que el cliente no desea que se generen archivos auxiliares (como un repositorio de imágenes), se emprendió una investigación para conocer las diferentes alternativas existentes para la inclusión de imágenes en un entorno JavaScript, que es el seleccionado teniendo en cuenta que es un requerimiento del cliente que se mantengan los mismo subsistemas del año pasado, basados en su parte de visualización en la librería especificada en el punto anterior.

Estas imágenes, según lo especificado por el cliente y los antiguos desarrolladores de la aplicación, deben asociarse a nuevos tipos de artefactos, que se crearán con este fin. De tal manera podrán asignarse un mismo tipo de artefacto a varios artefactos, que aunque distintos tendrán la misma representación. Un ejemplo bastante claro sería la presencia de resistencias en un circuito eléctrico. Pueden existir muchas, pero cada una puede tener a su vez diferentes propiedades, sin embargo, en un diagrama todas tendrán la misma representación al ser todos elementos de tipo resistencia.

A continuación se mostrarán las diferentes alternativas tecnológicas que se han barajado para solucionar este problema.

3.5.1 Serialización de las imágenes como atributos de archivos XML

La primera alternativa que surgió ante la negativa del cliente de añadir un repositorio de imágenes en los sistemas donde estuviera instalada esta aplicación fue añadir ficheros XML.

El lenguaje de marcas extensible XML (e**X**tensible **M**arkup **L**anguage) es un lenguaje que fue desarrollado por el World Wide Web Consortium (W3C) y que permite almacenar datos en documentos de manera estructurada, lo que permite a su vez dar soporte a bases de datos, parte fundamental para este proyecto.

```
<?xml version="1.0" encoding="UTF-8"?>
- <Root>
  - <Customers>
    - <Customer CustomerID="XYZ">
        <CompanyName>XYZ Market</CompanyName>
        <ContactName>John Doe</ContactName>
        <ContactTitle>Marketing Manager</ContactTitle>
        <Phone>(212) 555-1212</Phone>
        <Address>100 Park Avenue</Address>
        <City>New York</City>
        <State>NY</State>
      </Customer>
    - <Customer CustomerID="ABC">
        <CompanyName>ABC Market</CompanyName>
        <ContactName>Jane Doe</ContactName>
        <ContactTitle>VP Marketing</ContactTitle>
        <Phone>(201) 555-1213</Phone>
        <Address>102 Main Street</Address>
        <City>Newark</City>
        <State>NJ</State>
      </Customer>
  </Customers>
</Root>
```

Ilustración 8: Ejemplo de documento XML

Usando archivos XML, se permite comunicar el módulo de representación gráfica con la base de datos del Knowledge Manager (de donde toma todos los datos este módulo que se está mejorando) para pasarles las imágenes asignadas a los diferentes tipos de artefactos con representación creados por el usuario. A su vez, la base de datos a través de un elemento que funcione como un controlador, proporcionará los datos para representar las imágenes asociadas a los tipos de artefactos que se asignen a un diagrama en el módulo de representación gráfica creándose de nuevo archivos XML que a través de implementaciones JavaScript son leídos (al ser estructurados) obteniendo las imágenes asociadas.

A su vez, las imágenes desde sus diferentes formatos de origen (PNG, JPEG, TIFF…), pueden ser serializadas convirtiéndose en fragmentos de memoria conocidos como Memory Streams en programación, que son fácilmente serializables en una base de datos si son almacenadas de manera estructurada en estos ficheros XML. Estos fragmentos de memoria al deserializarse se convertirían en imágenes en un repositorio temporal, de tal manera que JavaScript pudiera

recogerlas para mostrarlas en el navegador, eliminándose al final de la ejecución del módulo este repositorio temporal.

Aunque es una solución factible, presenta algunas complicaciones que no son asumibles debido a la cantidad de esfuerzo que requiere implementarlas, teniendo en cuenta que además el rendimiento no será el idóneo. Esto es debido a que la acción de serializar y deserializar elementos en un documento XML, requiere un gasto computacional muy alto, sobretodo en función del tamaño de la imagen que se esté almacenando.

Además, se requieren muchos accesos a la base de datos al volcar el fichero XML, ante las diferentes modificaciones el usuario realice en los nuevos tipos de artefactos durante la ejecución del módulo.

Esto podría solucionarse mediante un modelo desconectado que vuelque los datos solo al final de la ejecución del módulo, que es el modelo que se sigue para el mismo Knowledge Manager, sin embargo, y ante la carga de numerosos diagramas por parte del usuario, no sería una solución factible. Mayormente debido a que hasta que se vuelquen todos estos ficheros en la base de datos, se quedarán almacenándose ocupando muchísimo espacio en el disco duro, lo que es un problema de rendimiento grave que no es asumible.

Sin embargo y gracias a esta primera aproximación, se llegó a la segunda alternativa para esta necesidad, que será la que finalmente se ha decidido implementar a lo largo del desarrollo de este proyecto.

3.5.2 Serialización de imágenes mediante Base64 Strings

La siguiente solución consiste en convertir las imágenes que se cargan en strings en Base64, ¿pero qué es este tipo de string?

Para empezar Base 64 es un sistema de numeración posicional que usa 64 como base, de tal manera que se codifican strings en los que se puede albergar cualquier tipo de dato usando esta notación. También es necesario conocer qué es un string, que no es más que una cadena de caracteres de manera ordenada que se recogen en este tipo de representación de datos.

Usando por lo tanto estos strings en Base 64, se pueden codificar imágenes que se almacenan en un formato mucho más cómodo para poder tratarlas. Desde este formato se pueden convertir a casi cualquier otro tipo de dato, como pueden ser fragmentos de memoria, arrays de Bytes y otros muchos más, que facilitan el soporte para diferentes bases de datos y diferentes sistemas.

Otra ventaja de usar estos tipos de strings, es que tienen un **soporte nativo en JavaScript,** lo que hace que al cargar uno de estos strings en un navegador, y marcarlo como un objeto de tipo imagen en JavaScript, se mostrará en el navegador como una imagen.

Así se ha decidido usar esta alternativa, de tal manera que las imágenes se recogerán en strings de Base 64 que se almacenarán como atributos de nuevos tipos de artefacto que cree el usuario, convirtiéndose al formato que necesite el cliente para almacenarse en la base de datos sin tener que realizar ninguna modificación en la misma salvo la de añadir un nuevo campo para estas representaciones, lo que no deja de ser una tarea realmente sencilla.

A continuación se muestra una imagen, y su representación en un fragmento de un string de Base 64.

/9j/4AAQSkZJRgABAQAAAQABAAD/2wCEAAkGBxQSDxAQEhIQEBUUDxAQFBAQDxQPFBQ
UFRQXFhQRFBQYHSggGB0lGxQVITEhJSkrLi4uFx8zODMsNygtLisBCgoKDg0OGhAQGi
wkICQsLCwsLCwsLCwsLCwsLCwsLiwsLCwsLCwsLCwsLCwsLCwsLCwsLCwsLCwsLCwsL
CwsLP/AABEIAOEA4QMBEQACEQEDEQH/xAAcAAEAAQUBAQAAAAAAAAAAAAAAABAEDBQYH
Agj/xABFEAABAwICBQkEBggFBQAAAAABAAIDBBEFIQYSExQxByJBUWFxgZGhUnKxwTJ
CYoLh8CMkM1OSstHxFRaTovIXNENEc//EABsBAQACAwEBAAAAAAAAAAAAAAACAwEEBQ
YH/8QANhEAAgIBAgQCCAUEAgMAAAAAAAECEQMEEgUhMVETQSIyYXGBkaHRBkKxwfAUM
1LhFfEjJGL/2gAMAwEAAhEDEQA/AO4oAgCAIAgCAIAgCAIAgCAIAgCAIAgCAIA
gCAIAgCAIAgCAIAgCAIAgKEoDA41ppQ0lxNVQtcP/GHh7/4G3Kkot9DDkjV5+V
hjzajoa+tPQY4HBp8QCfRS8PuS/sW4zHXPUAHxB1T6LO2HcbnXQ9f41pC
f/AE6NvYZvxWKh3FyPLsax9nHDqaX3Kho+LkqHcXLxLseP8P8AqHiEJ/WsFq2jpfBrTAeTL
f71nYn0Y3O+hkMO5WaF5Dx9ZXZXSUrvZqIyy3eeCi8cgpx3HD8WhnaHwyiqXa4+cp0kFixnB
omnZNQBAEAQBAEAQBAEAQBAEBRzrC5yQGj45zkFixna
+TgBdWLG6t8iDmuiMX/l7Eq4uIVu5u6XhsHFxdTbsHbzFndEPGBdPpedH
TMe+99rOTO+/Xd5Ib4AKLnXgEyetqgG2QG2QEjeH2QMjLJu
8MMwPESRtf6kIm0KRp1dyYi0iOjfEeAXH2xwdoPRBj2K4K
JtfA3jU0oGuG9b2WHw8VnYAOnugNzKAkoqqhOS3cP2NpcRQW4F3D12lIyFxyHrUc5EdV51Y6v0
seeqRg+I8bqan5Mg4+aLejmnstPMKLFGCGXgycfspR0EO4D88OCSx+cRHJzqR0uGUOA
LTcKotLiAIAgCAIAgCAIAgNd000uhw6nM9WfaHYP6ZBgW+4V
4VLVzjEMS58nGGld+zhBLT058PE58ShY3hBZXeEoWV3hKFjeEoWbW0zqjg5sahf
3hKFjeEEoWV3hKFjeEoWV3hKFjeEoWV3hjZjsdwyGsiM7Q4cwuy1mO9ph6C
sptdDSZq2j+kk+VIVxIRm4pHQy/BgbIwcXWV+pR5mmY+nSSqeMvupYJhSXqBXKSk9raI1
oRHLxWANuXFwiaKaWVk+B1hd4SfT+dUqsOZXHa1v9l5T0HqzLIVnn/DQzTlYx7ttsSSnwA6
hwcFJOhZGuEMWUMvHm6kEIcxy2W5Vk6V4HnvqRNsvPhE8BMYWJuDmPqspt1hL/pYqBlmhxD3tKf
f/AE6NvYZvxWKh3FyPLsax9nHDqaX3Kho+LkqHcXLxLseP8P8AqHiEJ/WsFq2jpfBrTAeTL
ZrQ5Iz28KFE7G8JQsbwlCxvCULG8JQsbwlCxvCYLLv7S7i7P

Ilustración 9: Imagen antes y después de la codificación en string de Base 64

3.6 Proceso de selección de algoritmos para dibujar diagramas de circuitos eléctricos

En este apartado se especificarán los diferentes algoritmos que se han estudiado para resolver el problema de dibujar circuitos eléctricos.

Dibujar circuitos eléctricos como mejora para este módulo es una tarea de una complejidad inmensa, que intenta abarcar en lo posible todas las formas en las que se representan los circuitos eléctricos.
Cabe destacar además, la necesidad de asistencia del cliente para especificar todos los casos que quiere contemplar a la hora de dibujar estos diagramas.

A continuación se expondrán las diferentes formas que adquirirán los diagramas eléctricos según sus necesidades.

3.6.1 Configuraciones de circuitos eléctricos según las necesidades del cliente

En un principio se estipuló el desarrollo de dos tipos de diagrama de circuito eléctrico, por un lado los circuitos en serie/paralelo que no forman un ciclo al cerrar el circuito y los que sí. Sobre estos últimos, además se especificó la distribución de los elementos del diagrama según las especificaciones del cliente.

A continuación se irán enumerando los casos con imágenes que permitan entender mejor los mismos.

3.6.1.1 Circuitos en serie/paralelo en los que no se cierra el circuito

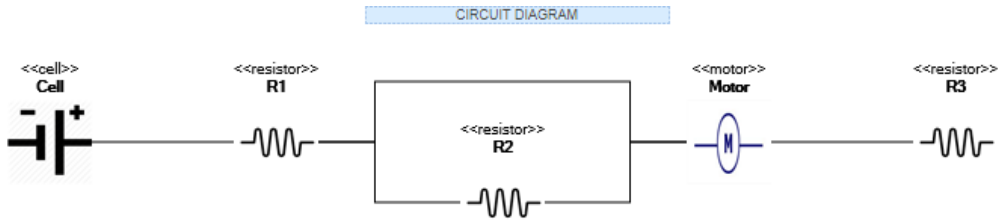

Ilustración 10: Circuito eléctrico serie/paralelo sin cerrar

En este tipo de circuitos, la disposición de los elementos que conforman el diagrama, así como de las relaciones que unen estos, es la de un eje horizontal con todos estos elementos distribuidos en él. La imagen anterior muestra este caso.

3.6.1.2 Circuitos en serie/paralelo cerrados

En este casos se contemplarán tres opciones:

- **El primer elemento y el último no preceden/van seguidos de ningún elemento en paralelo:** En este caso tanto el primer elemento como el último se colocarán en vertical según el convenio establecido por el cliente.

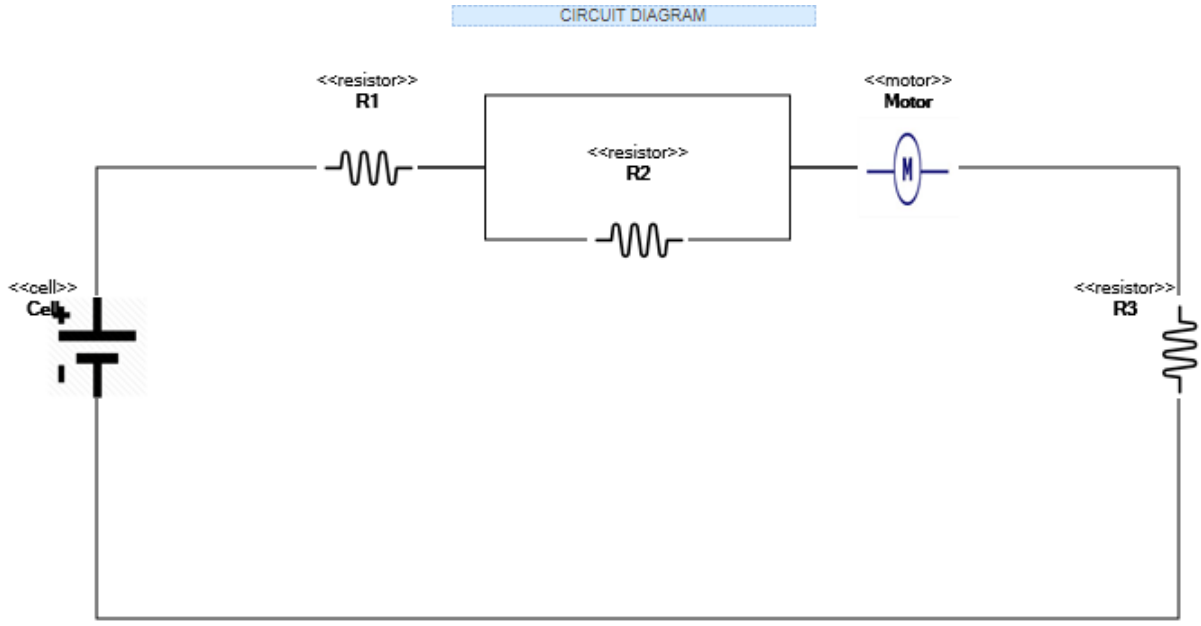

Ilustración 11: Circuito eléctrico serie/paralelo cerrado, primer caso

- **El primer elemento precede a elementos en paralelo, mientras que el último no tiene ninguna relación entrante de un elemento en paralelo:** En este caso el primer elemento del circuito no se pone en vertical, sino que se mantiene en el eje horizontal, sin embargo el último si se mantiene.

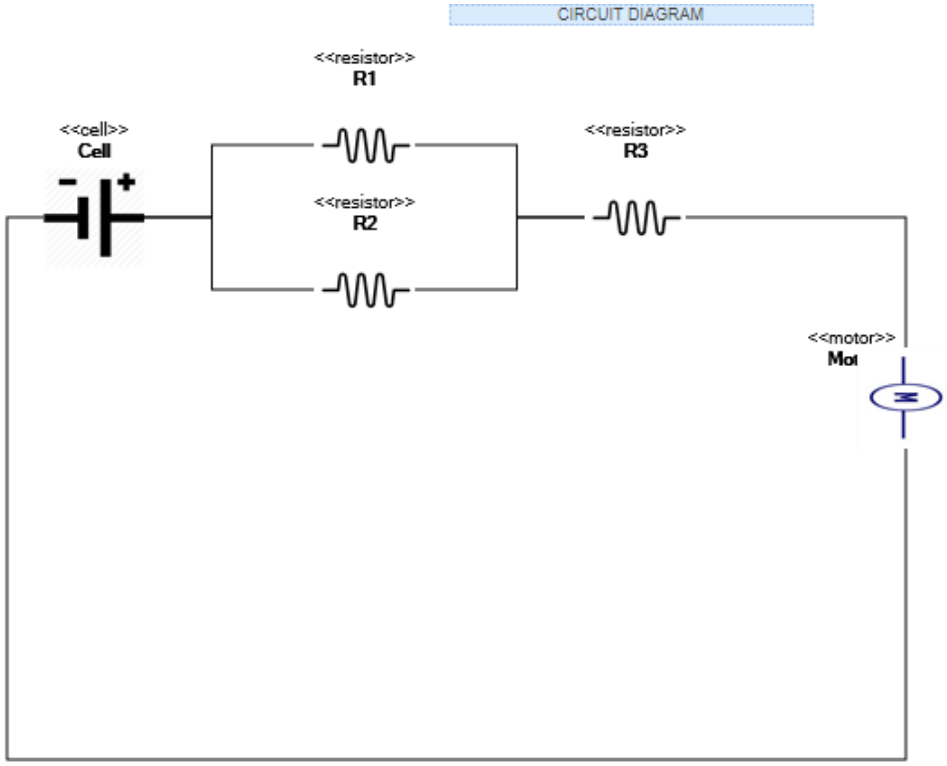

Ilustración 12: Circuito eléctrico serie/paralelo cerrado, segundo caso

- **El primer elemento no precede a elementos en paralelo, mientras que el último sí tiene alguna relación entrante de un elemento en paralelo:** En este caso el primer elemento si se mantiene en vertical, mientras que el último elemento se mantiene en horizontal debido a que le preceden elementos en paralelo.

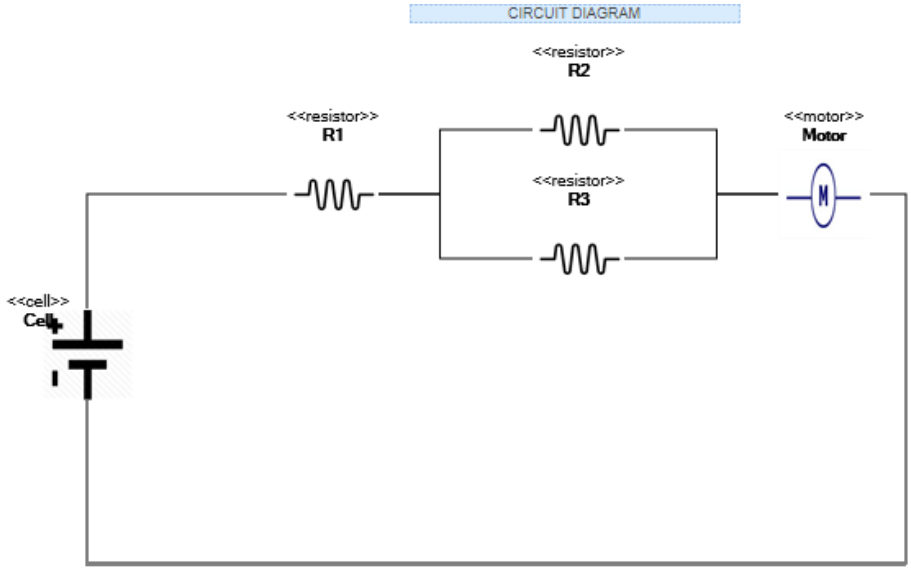

Ilustración 13: Circuito eléctrico serie/paralelo cerrado, tercer caso

Con este último caso se delimitan las necesidades en la disposición de los elementos del circuito.

Así para pintar estas situaciones, contemplando los casos especiales en los que baja el primer elemento o el último se ha modificado una función presente en la librería gráfica que mediante un algoritmo que se aplica en todos los elementos, crea un grafo dirigido (es decir con dirección, o mejor dicho, sentido entre los nodos, no es igual ir de A a B que de B a A) de manera recursiva aplicando el siguiente algoritmo:

- Se establecen dos vértices, uno que funciona como origen y otro como destino unidos por una relación considerándose un grafo dirigido en forma serie/paralela. Si más grafos dirigidos tomados de dos en dos (como el anterior) son grafos series/paralelos lo serán por dos operaciones:
    - Composición paralela: Comprobándose que el origen del primer grafo dirigido es el mismo que el del segundo grafo dirigido, y sus finales son iguales también.
    - Composición en serie: Comprobándose que el final del primer grafo dirigido es el origen del segundo grafo dirigido.

Realizando estas comprobaciones se puede saber si el diagrama corresponde a un diagrama con todos sus elementos dispuestos en serie o en paralelo. En caso de que haya ciclos, por ejemplo cuando se cierra un circuito, este algoritmo reconocerá que es un diagrama que no es serie/paralelo. Por esta razón se han barajado diferentes alternativas para permitir ciclos en el diagrama, como por ejemplo, el cierre de un circuito.

3.6.2 Selección de algoritmos para detectar ciclos

Siguiendo con la línea en la que terminó el apartado anterior, una situación muy complicada que se plantea es como detectar ciclos, además de distinguir que el diagrama forma un grafo dirigido con una configuración serie/paralela utilizando el algoritmo también definido en el apartado anterior. Por tanto lo que se intenta localizar es lo que se denominan **Componentes fuertemente conexas** (En inglés Strongly Connected Components) que no son más que la definición de los ciclos que se están intentando buscar. Así la definición de estos componentes fuertemente conexos es la siguiente:

- Un grafo dirigido llamado fuertemente conexo, es aquel que para cada par de vértices u y v, existe un camino de u hacia v y un camino de vuelta de v hacia u.

Para ello se han barajado los siguientes algoritmos con el fin de encontrar estos ciclos:

- **Búsqueda en profundidad:** La búsqueda en profundidad es un algoritmo que recorre los nodos de un grafo de manera ordenada, de tal manera que va expandiendo todos los nodos que va localizando formándose así un camino concreto. Cuando no quedan más nodos que visitar, retrocede desrecorriendo el camino y repitiendo el proceso

hasta que no haya más tramos posibles. De esta manera el algoritmo es capaz de encontrar los ciclos que sólo con el algoritmo anterior no se permitía.
Este algoritmo tiene un tiempo de ejecución aceptable ya que su complejidad depende del número de vértices más el número de aristas que tiene el grafo, siendo esta lineal, por lo que su aplicación sería aceptable. Sin embargo, este algoritmo permite localizar ciclos, ya que detecta cuando se debe retroceder a un nodo que ya se había recorrido, pero no permite identificarlos y separarlos, es decir, el algoritmo sabe que existe al menos un ciclo, pero no sabe cuántos ni cómo se distribuyen, por lo que su uso queda limitado y por tanto queda descartado como alternativa válida.

Por ejemplo, si se aplica este algoritmo, puede detectarse un ciclo cuando se cierra el circuito, pero ¿y si además hay una realimentación el circuito? No se podría detectar todas las relaciones implicadas en la misma si es compleja (es decir varios elementos conectados en serie que se realimentan en el circuito) y por lo tanto el algoritmo que se encarga de saber si este diagrama puede ser un diagrama serie/paralelo, no lo distinguiría como tal degenerando en un problema en la funcionalidad del sistema que no se puede permitir.

- **Algoritmo de Kosaraju:** Este algoritmo se basa en el hecho de que si se invierten todas las aristas de un grafo, las componentes fuertemente conexas son las mismas que en el grafo original. Con lo cual se pueden detectar ciclos y distinguirlos para poder tratarlos adecuadamente de manera individual.
  Los pasos necesarios definidos para aplicar este algoritmo se muestran en la siguiente cita (Algoritmos de Grafos para Componentes Fuertemente Conexas, 2014):

  1. Realiza una búsqueda en profundidad sobre G y enumera los vértices.
  2. Construye un nuevo grafo G' invirtiendo la dirección de todo los arco en G.
  3. Realiza una búsqueda en profundidad sobre G' empezando por el vértice que fue enumerado con el mayor valor de acuerdo a la numeración asignada en el paso (1). Si la búsqueda en profundidad no alcanza todos los vértices, empieza la siguiente búsqueda en profundidad desde el vértice restante enumerado con el mayor valor.
  4. Cada árbol obtenido luego de la búsqueda en profundidad es una componente fuertemente conexa.

  También de esta fuente se ha obtenido una imagen que explica el proceso de manera sencilla.

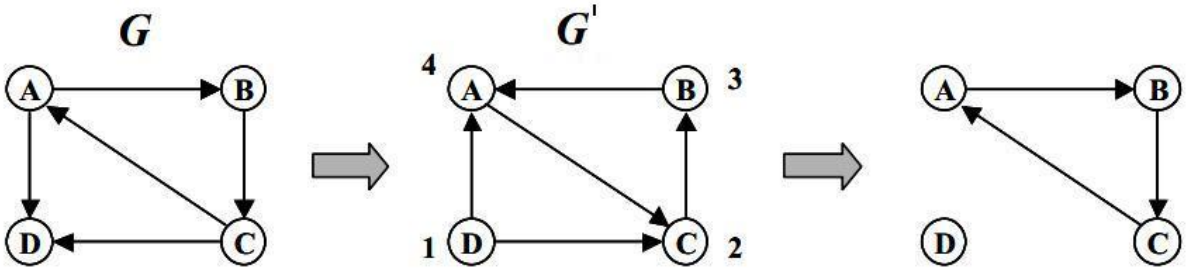

Ilustración 14: identificación de una componente fuertemente conexa con el Algoritmo de Kosaraju paso a paso

Este algoritmo por lo tanto permite detectar ciclos en un diagrama, distinguiéndolos de tal manera que pueden tratarse de manera independiente, por lo que si se combina este algoritmo con el algoritmo que permite la configuración de circuitos en serie/paralelo se cubren todas las necesidades del cliente.
Su tiempo de ejecución es óptimo ya que tiene una complejidad lineal siendo esta la suma del número de vértices más el número de aristas del grafo. Sin embargo, se requieren dos ejecuciones de la función que recorre los nodos del grafo aplicando una búsqueda en profundidad (una para el grafo dirigido original, y otra para su grafo dirigido invertido), por lo que aunque el tiempo de ejecución es bueno, requiere una implementación algo complicada al requerir bastantes estructuras y funciones con los que comprobar el grafo dirigido original con su grafo dirigido invertido.

Por lo tanto esta alternativa también queda descartada en favor de la tercera alternativa que se muestra a continuación.

- **Algoritmo de Tarjan:** Este algoritmo no deja de ser una versión mejorada de la búsqueda de componentes fuertemente conexas que realiza el algoritmo de Kosaraju, pero en lugar de tener que recorrer el gráfico dirigido invertido del original para encontrar los ciclos, usa una pila para realizar la misma función. La ejecución del mismo es la siguiente:

  1. Se realiza una búsqueda en profundidad sobre el grafo dirigido, creándose una lista en la que se van añadiendo los vértices mientras se quedan sin vecinos que no hayan sido visitados.
  2. Se traspone el grafo de dirigido.
  3. Se realiza una búsqueda en profundidad eligiendo los vértices en orden contrario en el que aparecen en la lista creada en el punto 1.

  Como se puede apreciar, este algoritmo cubre las mismas necesidades que el algoritmo de Kosaraju, pero su implementación es infinitamente más sencilla, por lo que será la alternativa seleccionada para encontrar ciclos en los diagramas de circuitos eléctricos.

Así la opción elegida finalmente será el algoritmo de Tarjan.

4. Análisis

En este punto se define la aplicación desarrollada al completo, con todas las funcionalidades que debe cumplir junto a su alcance. El objetivo de esta fase consiste en detectar las necesidades del problema que se debe resolver, de tal manera que se pueda diseñar el sistema para solventarlas posteriormente.

4.1 Definición del sistema

En este apartado de este documento se describirá el sistema, de tal manera que se puedan detallar los problemas que se resuelven para poder realizar un análisis adecuado.

4.1.1 Determinación del alcance del sistema

A continuación se describirá cual son las principales funciones que debe realizar le módulo que se está desarrollando.

En este trabajo de fin de grado como se ha descrito anteriormente se pretende implementar una mejora en la herramienta asociada al Knowledge Manager que permite representar gráficos. Basado en RSHIP como modelo de datos, pretende representar cualquier tipo de información que sea contenido en el modelo de datos.

Pero este módulo que se está desarrollando se centrará en implementar una mejora principalmente en la parte la interfaz gráfica, así como en los subsistemas que actúan como controlador y modelo (siguiendo la arquitectura Modelo-Vista-Controlador), de tal manera que se muestre una representación gráfica de una serie de datos transformados desde RSHIP, que será el lenguaje nativo en el que se representará originalmente esta información, mostrándolos con la filosofía de dibujo en la que el usuario quiera mostrar está información. Es decir pintar la información que originalmente estaba representada con los elementos colocados con las filosofía de dibujo de RSHIP (muy parecido a como se representaría un gráfico UML) para poder representar esa misma información con otra disposición gráfica más adecuada a lo que realmente se quiere representar, por ejemplo, si se representan elementos de un circuito eléctrico conectados en RSHIP, este módulo ofrece al usuario recalcular la disposición de estos elementos y de sus conexiones con la forma de un circuito eléctrico, incluso representando los iconos de los diferentes elementos contenidos en ese gráfico en el modo transformado.

A continuación se muestra una imagen que representa perfectamente la comunicación del módulo con el KM, y que fue creada por mi compañero Pablo Sánchez Pérez. También se puede apreciar como el módulo que vamos a mejorar se basa en tres funciones principales como detallo también mi compañero Pablo el año pasado.

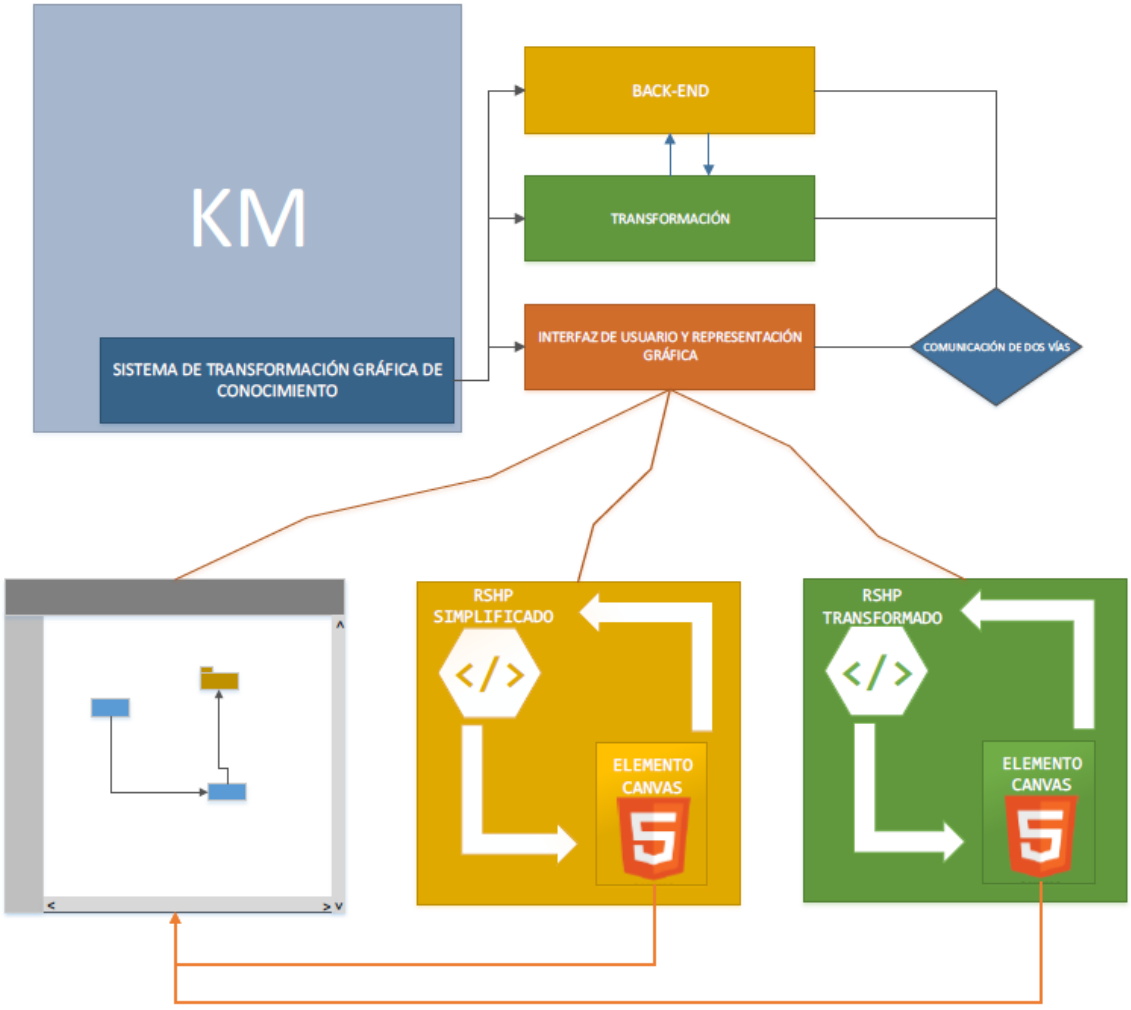

Ilustración 15: Diagrama de alcance de sistema creado por Pablo Sánchez Pérez

Como mencionó, este módulo tiene tres funciones principales:

- ***Crear la interfaz gráfica:*** *Diseñar e implementar el entorno donde el usuario pueda visualizar y editar los elementos gráficos. En esta interfaz el usuario puede crear un modelo nuevo, estos cambios tendrán que ser notificados al* ***back-end****.*

- ***Crear los elementos gráficos a partir de RSHP [1]simplificado:*** *El subsistema encargado de realizar un procesado del RSHP que maneja el KM y simplificarlo, establecerá una comunicación con nuestro subsistema y enviándolo a este módulo. Este proyecto es el encargado de transformar esta información a elementos gráficos que deberán ser añadidos a la interfaz de usuario. Este subsistema también puede manipular el modelo, con lo que se establecerá una nueva comunicación a la inversa.*

- ***Crear los elementos gráficos a partir de RSHP transformado:*** *Esta tarea no es más que una extensión de la anterior en lo que a comunicación se refiere. El sistema ha de ser capaz de pasar de un modo nativo (representación gráfica del RSHP simplificado) a otro modo de transformación (representación gráfica del RSHP transformado). La diferencia principal entre estos dos modos es que, en este último, el usuario podrá modificar las propiedades de los elementos transformados (con lo que comunicará con el subsistema de transformación) pero no añadir elementos nuevos.*

Esta mejora como se ha mencionado en repetidas ocasiones va a trabajar principalmente sobre la última funcionalidad que se detalla en la cita de la memoria de Pablo Sánchez Pérez, es decir, **crear los elementos gráficos a partir de RSHIP transformado.** El alcance de esta mejora quedará definido en los requisitos.

4.1.2 Identificación del Entorno Tecnológico

El módulo que se va a mejorar seguirá desarrollándose en el mismo entorno de programación que el usado en el desarrollo del Knowledge Manager, este IDE es el que se ha mencionado anteriormente y es el Visual Studio 2010. Además teniendo en cuenta todas las necesidades de hardware que tienen el resto de aplicaciones que se usarán en el desarrollo de este módulo (control de versiones, Office, etc.). Así el entorno mínimo será el siguiente:

- Microsoft .NET Framework 4.0.
- Procesador a 1.5 GHz.
- 512 MB de memoria RAM.
- 64 GB de disco duro.
- Sistema Operativo: Windows Vista, Windows 7, Windows 8 y Windows 8.1.

[1] RSHP: Era el anterior nombre que recibía este modelo de datos, pero como era difícil de pronunciar derivó en RSHIP.

4.1.3 Especificación de estándares y normas

En este apartado de análisis del sistema, ha sido necesario cumplir con ciertos estándares para obtener un producto con cierta calidad.

Todas las salidas de cada una de las fases de este documento están especificadas en Métrica V3, que configura el estándar usado en el desarrollo de la documentación para este proyecto de fin de grado, pero esto no significa que se cumplan todas las fases descritas en el mismo, sino sólo las que el desarrollador considere necesarias para este proyecto.

4.1.3.1 Restricciones generales

Otras restricciones que se deben cumplir para el subsistema se encuentran principalmente en la comunicación con los otros subsistemas que configuran el módulo desarrollado por mis compañeros del año pasado.

La primera consiste en que las mejoras implementadas sobre el módulo deben ser compatibles con al menos la versión de Internet Explorer 7. Esto se debe a que las mejoras implementadas para este módulo deben funcionar dentro de la arquitectura con la que ya cuenta el mismo, ya que el módulo se ejecuta en el KM dentro de un elemento llamado Web Browser que ejecuta una versión de este navegador, y permite cargar todos los elementos web que se usan en la representación de la información.

La segunda consiste en que cualquier modificación dentro de la programación de los elementos que se muestran deben ser instancias siempre de las clases Node y Link de la librería Lassalle. Esta restricción hace referencia a la que Pablo Sánchez Pérez estableció el año pasado durante su investigación de la librería Lassalle, por lo que es interesante mencionarlo también en este documento.

4.1.3.2 Supuestos y Dependencias

Este proyecto de fin de grado al consistir en una mejora de un proyecto que ya estaba desarrollado tiene una dependencia clara con el desarrollo anterior. Esto quiere decir que cuando se realice cualquier cambio en el subsistema que asienta la base de la mejora que se está desarrollando, se debe tener en cuenta y aplicar los cambios necesarios, revisando la mejora desarrollada para que pueda adaptarse a estas modificaciones.

4.1.3.3 Estándar de código

El estándar de código usado en el desarrollo de este proyecto se define con los siguientes puntos:

- Cada función desarrollada debe ser comentada con su contenido, además de una breve descripción de la misma, su entrada y su salida.
- Se usará el estándar de sangrías usado por el equipo de The Reuse Company, que viene dado en un archivo de configuración que se instala en el Visual Studio 2010.
- Para el sangrado de JavaScript se usará un plugin descargado para Visual Studio 2010: JScript Editor Extensions.

4.1.3.4 Entorno operacional

El entorno operacional que se usará en el desarrollo de este proyecto de fin de carrera estará conformado por:

- Sistema Operativo Windows 8.1 64-bit
- Microsoft Office 2013, incluido las aplicaciones Project y Visio.
- Tortoise SVN 1.7.8 64-bit
- Java versión 7 Update 67
- .NET Framework 4.0

4.1.4 Identificación de los usuarios participantes y finales

En este apartado se identificarán los usuarios participantes en el análisis del sistema, teniendo en cuenta además a los usuarios que validan y aceptan la aplicación.

Es importante en este apartado identificar a las personas físicas y organizaciones (Stakeholders) que estén involucrados en este proyecto:

- **Jefe de Proyecto:** Juan Bautista Llorens Morillo.
  Será el tutor de este proyecto de fin de grado y el Jefe de proyecto dentro de la compañía The Reuse Company. Será la persona encargada de establecer las necesidades del proyecto y de tomar la decisiones finales que afecten al mismo.

- **Usuario Final**:
  El usuario final de la herramienta KM. Puede ser experto en el manejo de software de gestión del conocimiento o no.

- **The Reuse Company:**
  Empresa cliente de esta mejora que se está desarrollando para incluir estas modificaciones en el módulo de representación gráfica existente dentro de sus software Knowledge Manager.

- **Universidad Carlos III de Madrid:**
  Entidad pública donde se presenta el proyecto como trabajo de fin de Grado.

4.2 Identificación de subsistemas de análisis

Para llegar a comprender la funcionalidad que ofrece este nuevo módulo, se va a dividir este en diferentes subsistemas, cada uno con una función dentro del módulo que hace que se pueda comprender por sí sólo. Esta división en subsistemas permite identificar subsistemas que compartan procesos o afinidad en los requisitos.

Estos subsistemas se basan en los que se elaboraron el año pasado en la parte de análisis del módulo en la etapa anterior de su desarrollo, aunque se incluirán algunas modificaciones mostrando las funciones nuevas que el sistema proporciona al finalizar esta etapa de desarrollo:

- **Modificaciones en los subsistemas la capa de vista:**

  - **Barra de herramientas:** En la barra de herramientas existente, que cuenta con una serie de acciones que se aplicarán sobre el diagrama representado, se han implementado las siguientes mejoras:
    - **Añadir nuevos tipos de artefacto:** Permitiendo definir nuevos tipos de artefacto desde este módulo sin tener que usar el Knowledge Manager. Además se pueden añadir representaciones en forma de iconos al nuevo tipo de artefacto, junto con la orientación de la imagen para poder representarla correctamente en el elemento del diagrama que contenga este tipo de artefacto. Estos nuevos tipos de artefactos quedarán serializados en el modelo de datos una vez sean añadidos de igual manera que se realizaría en el Knowledge Manager.
    - **Acción de layout:** Esta opción permite, que tanto en modo nativo como en transformado, se aplique la filosofía de dibujo concreta que se ha definido para el diagrama en cuestión.
    - **Definir el layout del diagrama que se está representando:** Esta acción define el tipo de filosofía de dibujo que tiene el diagrama, de tal manera que se pueda aplicar la opción de layout.
  - **Representación gráfica transformada:** Se encarga de representar los elementos que estaban originalmente en su forma de representación nativa (representados mediante el modelo RSHIP) en su forma transformada. La forma transformada de un diagrama depende de la filosofía de dibujo que se quiere representar por un lado (circuito eléctrico, diagrama UML o diagrama de secuencia), así como de si sus elementos tienen representación (una imagen, un color o forma concretos, etc.).
  - **Interacción del usuario con los elementos del Canvas:** Permite al usuario gestionar la creación y edición del diagrama y de los elementos que lo conforman. En este subsistema las modificaciones que se han realizado hacen referencia a opciones de configuración en lo elementos, relacionadas con el tipo de filosofía de dibujo que tiene el diagrama. Un ejemplo sería la selección del primer elemento para identificarlo en ciclos cuando se pretende invocar la acción de layout para el tipo de diagrama de circuito eléctrico.
  - **Paleta de edición:** Será la encargada de proporcionar al usuario los elementos que puede añadir al diagrama arrastrándolos. Las mejoras implementadas en este

subsistema consisten en limitar los elementos que se pueden añadir al diagrama en función del tipo que sea.

- **Modificaciones en la capa de controlador:**

  - **Subsistema de Nodo:** Es el módulo que recoge los datos de la capa de la vista acerca de los nodos que se muestran en el diagrama. En este módulo se han añadido las opciones de recoger representaciones de los mismos, así como algunos atributos nuevos y necesarios para controlar la disposición de los nodos según el tipo de filosofía de dibujo con la que se esté trabajando en el diagrama.
  - **Subsistema de gráfico:** Contiene la funcionalidad del sistema casi por completo, y dentro de la capa de controlador, es el subsistema que funciona casi como disparador de todas las acciones dentro de esta capa de la arquitectura del proyecto. En este subsistemas se han implementado modificaciones relacionadas con la lógica de dibujo de circuito eléctrico, la representación de manera correcta de las imágenes dentro de los nodos (si las tienen), el giro de estas imágenes según la orientación de las relaciones que estén conectadas a ese nodo, reconocimiento de ciclos de retroalimentación en circuitos eléctricos para cambiar el sentido de las relaciones de los mismos y del sentido de la imagen dentro del nodo…

4.3 Establecimiento de requisitos

En este punto se van a presentar los requisitos software que son necesarios para el módulo que se está desarrollando, estos se han recogido en reuniones periódicas con el cliente en diferentes momentos, aunque sobretodo en la fase inicial del proyecto.
El proceso de obtención de todos los requisitos para este proyecto de fin de grado comienza con los requisitos de alto nivel, es decir, aquellos que son requisitos de necesidad o requisitos de los stakeholder (se han recogido en el punto 3.2), y luego estos se irán desgranando en requisitos más específicos, llegando a los requisitos de sistema, que detallan el sistema basándose en los requisitos de nivel superior.

Una vez definidos los requisitos, es interesante recoger casos de uso relacionados para probar el entorno de sistema, todo esto se recogerá más adelante.

4.3.1 Obtención de requisitos del sistema

Los requisitos del sistema se encargan de definir que debe cumplir el sistema para que las necesidades que definió el usuario queden cubiertas, siguiendo el proceso que se definió en la introducción anterior.

Los requisitos se recogerán en tablas como la que se muestra a continuación, justamente debajo se especifican los diferentes campos con los que cuenta.

ID: RS-XYY					
Fecha	Dd/mm/aaaa	**Tipo**	Tipo del requisito		
Fuente	Fuente de la que procede	**Estado**	Propuesto		Cerrado
Necesidad	Esencial	Deseable		Opcional	
Prioridad	Alta	Media		Baja	
Complejidad	Alta	Media		Baja	
Objetivo	Objetivo que pretende conseguir el requisito.				
Precondiciones	Precondiciones a cumplir por el requisito para ser llevado a cabo (si existen).				
Descripción	Descripción del requisito.				
Trazabilidad	Trazabilidad del requisito que permite conocer su relación con otros requisitos.				

Tabla 22: Ejemplo de tabla de requisito de sistema

El identificador dota al requisito de un nombre unívoco, que lo distingue del resto. Teniendo en cuenta la estructura del identificador se pueden distinguir las siguientes partes:

- **RS**: Requisito de Sistema.
- **X**: Tipo de requisito, que se incluye también en la tabla. Dentro de los tipos de requisitos que se recogerán en este apartado encontramos:
  - **F:** Requisitos de tipo funcional.
  - **NF:** Requisitos no funcionales.
  - **S:** Requisito de seguridad.
  - **In:** Requisito de Interfaz.
- **YY:** Número identificador para requisitos del mismo tipo.
- **Prioridad, Necesidad y Complejidad:** Adquieren tres valores, alta media o baja.
- **Estado:** Este campo cuenta con los siguientes dos valores:
  - **Propuesto:** Se considera que un requisito se encuentra en estado propuesto cuando aún no ha podido verificarse, por ejemplo, al encontrarse en la fase de análisis.
  - **Cerrado:** Estarán en estado cerrado los requisitos que después de la fase de diseño del mismo estén cubiertos y probados.

4.3.1.1 Requisitos Funcionales

ID: RS-F001 – Crear nuevo tipo de artefacto					
Fecha	26/02/2015	**Tipo**	Funcional		
Fuente	The Reuse Company	**Estado**	Propuesto		Cerrado
Necesidad	Esencial	Deseable		Opcional	
Prioridad	Alta	Media		Baja	
Complejidad	Alta	Media		Baja	
Objetivo	El usuario debe poder crear nuevos tipos de artefactos que puedan contener una representación gráfica en forma de icono con un tamaño limitado.				
Precondiciones	No debe existir ningún tipo de artefacto con el mismo nombre contenido en la base de datos.				
Descripción	El sistema será capaz de permitir al usuario añadir un nuevo tipo de artefacto, permitiéndole definir mediante un panel el nombre del tipo de artefacto, la imagen que lo representará (si la tiene) con un tamaño limitado, la orientación de la imagen, y si este tipo de artefacto puede ser un diagrama definiendo también su filosofía de dibujo.				
Trazabilidad	RU-C01, RU-C02, RU-C13, RU-R02				

Tabla 23: RS-F001 - Crear nuevo tipo de artefacto

ID: RS-F002 – Editar artefacto existente para darle una nueva representación					
Fecha	26/02/2015	**Tipo**	Funcional		
Fuente	The Reuse Company	**Estado**	Propuesto		Cerrado
Necesidad	Esencial	Deseable		Opcional	
Prioridad	Alta	Media		Baja	
Complejidad	Alta	Media		Baja	
Objetivo	El usuario podrá editar los artefactos del diagrama para poder añadirle una representación (icono).				
Precondiciones	El usuario debe haber añadido al menos un artefacto al diagrama. Debe existir al menos un tipo de artefacto con representación en la base de datos.				
Descripción	El sistema asignará al artefacto que esté editando el usuario, el tipo de artefacto que seleccione el usuario, permitiéndole así dotarle de una representación gráfica.				
Trazabilidad	RU-C01, RU-C02, RU-C13				

Tabla 24: RS-F002 – Editar artefacto existente para darle una nueva representación

ID: RS-F003 – Sólo se mostrará la imagen asociada a un tipo de artefacto en el modo transformado					
Fecha	26/02/2015	**Tipo**	Funcional		
Fuente	The Reuse Company	**Estado**	Propuesto		Cerrado
Necesidad	Esencial	Deseable		Opcional	
Prioridad	Alta	Media		Baja	
Complejidad	Alta	Media		Baja	
Objetivo	El usuario sólo podrá ver la representación de los tipos de artefacto en el modo de transformación del módulo.				
Precondiciones	El usuario debe haber añadido al menos un artefacto al diagrama. Debe existir al menos un tipo de artefacto con representación en la base de datos. Un artefacto debe haberse editado seleccionando un tipo de artefacto con representación.				
Descripción	El sistema representará en el modo transformado la representación del tipo de artefacto asignado a un artefacto, siempre que la filosofía de dibujo que tiene el diagrama permita representaciones de este tipo.				
Trazabilidad	RU-C01, RU-C02, RU-C03, RU-C06, RU-C13, RU-R02				

Tabla 25: RS-F003- Sólo se mostrará la imagen asociada a un tipo de artefacto en el modo transformado

ID: RS-F004 – Los nuevos tipos de artefactos se podrán almacenar en la base de datos					
Fecha	26/02/2015	**Tipo**	Funcional		
Fuente	The Reuse Company	**Estado**	Propuesto		Cerrado
Necesidad	Esencial	Deseable		Opcional	
Prioridad	Alta	Media		Baja	
Complejidad	Alta	Media		Baja	
Objetivo	El usuario sólo podrá guardar los nuevos tipos de artefactos con todas sus propiedades en la base de datos.				
Precondiciones	El usuario debe haber añadido al menos un artefacto al diagrama. Debe existir al menos un tipo de artefacto con representación en la base de datos.				
Descripción	El sistema almacenará en la base de datos los nuevos tipos de artefacto creados por el usuario, así como todas sus propiedades asociadas (icono, orientación, filosofía de dibujo…) de igual modo que el usuario puede crear y guardar un nuevo tipo de artefacto dentro de la propia aplicación del Knowledge Manager. Las imágenes no se almacenarán de ningún modo como archivos locales.				
Trazabilidad	RU-C01, RU-C02, RU-C05, RU-C13, RU-I02				

Tabla 26: RS-F004 – Los nuevos tipos de artefactos se podrán almacenar en la base de datos

ID: RS-F005 – Las representaciones de los nodos podrán rotar					
Fecha	26/02/2015	**Tipo**	Funcional		
Fuente	The Reuse Company	**Estado**	Propuesto		Cerrado
Necesidad	Esencial	Deseable		Opcional	
Prioridad	Alta	Media		Baja	
Complejidad	Alta	Media		Baja	
Objetivo	Las representaciones contenidas en cada nodo (icono) podrán rotar y trasladarse los grados que sean necesarios para que esté orientado con la filosofía de dibujo del diagrama, siempre que esta filosofía esté basada en elementos conectados.				
Precondiciones	El usuario debe haber añadido al menos un artefacto al diagrama. Debe existir al menos un tipo de artefacto con representación en la base de datos y con una orientación definida por el usuario. Un artefacto debe haberse editado seleccionando un tipo de artefacto con representación.				
Descripción	El sistema rotará y trasladará los iconos que están contenidos en los nodos en el modo de transformación según la orientación del eje que forman las relaciones a las que esté conectado. Esto sólo ocurrirá en filosofías de dibujos basadas en elementos conectados, como por ejemplo la de circuito eléctrico, y en el modo transformado.				
Trazabilidad	RU-C10, RU-C12, RU-C13				

Tabla 27: RS-F005 - Las representaciones de los nodos podrán rotar

ID: RS-F006 – Creación de nuevos layouts					
Fecha	26/02/2015	**Tipo**	Funcional		
Fuente	The Reuse Company	**Estado**	Propuesto		Cerrado
Necesidad	Esencial	Deseable		Opcional	
Prioridad	Alta	Media		Baja	
Complejidad	Alta	Media		Baja	
Objetivo	El usuario podrá crear nuevos layouts para sus diagramas mediante la creación de nuevos tipos de artefacto.				
Precondiciones	-				
Descripción	El sistema ofrecerá al usuario la posibilidad de crear tipos de artefacto nuevos que sean considerados como layouts con una filosofía de dibujo concreta asociada al mismo. El sistema limitará la selección de layouts al usuario a tres: diagrama UML, diagrama de secuencia y diagrama de circuito eléctrico. Este límite viene impuesto por la necesidad de seguir desarrollando el resto de layouts antes de que se puedan ofrecer al usuario. Sin embargo el usuario podrá crear nuevos tipos de layouts seleccionando cualquiera de las filosofías de dibujo definidas en el módulo. Estás son circuito eléctrico, estructural y UML-De Secuencia.				
Trazabilidad	RU-C04, RU-C05, RU-C11, RU-C13				

Tabla 28: RS-F007 -Creación de nuevos layouts

ID: RS-F007 – Editar layout asociado al diagrama					
Fecha	26/02/2015	**Tipo**	Funcional		
Fuente	The Reuse Company	**Estado**	Propuesto		Cerrado
Necesidad	Esencial	Deseable		Opcional	
Prioridad	Alta	Media		Baja	
Complejidad	Alta	Media		Baja	
Objetivo	El usuario podrá seleccionar el tipo de layout que necesite para su diagrama entre los tipos de artefacto que existen en la base de datos.				
Precondiciones	Debe existir al menos un tipo de artefacto con layout en la base de datos.				
Descripción	El sistema ofrecerá al usuario la posibilidad de cambiar el tipo de filosofía de dibujo (layout) que tiene su diagrama, mediante el cambio del tipo de artefacto que está asociado al diagrama. El sistema sólo mostrará al usuario los tipos de artefactos que cuenten con un layout asociado.				
Trazabilidad	RU-C01, RU-C02, RU-C03, RU-C04, RU-C11				

Tabla 29: RS-F007: Editar layout asociado al diagrama

ID: RS-F008 – Límites en la selección de filosofías de dibujo					
Fecha	26/02/2015	**Tipo**	Funcional		
Fuente	The Reuse Company	**Estado**	Propuesto		Cerrado
Necesidad	Esencial	Deseable		Opcional	
Prioridad	Alta	Media		Baja	
Complejidad	Alta	Media		Baja	
Objetivo	El usuario sólo podrá crear tantos tipos de nuevos layouts como quiera, siempre que estos tengan distinto nombre y tengan asociado una de las filosofías de dibujo que permite usar el módulo.				
Precondiciones	-				
Descripción	El sistema ofrecerá al usuario la posibilidad de crear tantos tipos de layout como quiera, siempre que estos tengan distinto nombre y cumplan las condiciones estipuladas en el requisito RS-F006.				
Trazabilidad	RU-C11, RU-C12, RU-C13				

Tabla 30: RS-F008 – Límites en la selección de filosofías de dibujo – II

ID: RS-F009 – El sistema debe diferenciar los layouts					
Fecha	26/02/2015	**Tipo**	Funcional		
Fuente	The Reuse Company	**Estado**	Propuesto		Cerrado
Necesidad	Esencial	Deseable		Opcional	
Prioridad	Alta	Media		Baja	
Complejidad	Alta	Media		Baja	
Objetivo	El sistema ante la acción de layout por parte del usuario, debe distinguir la filosofía de dibujo que está asignada al diagrama.				
Precondiciones	-				
Descripción	El sistema debe reordenar los elementos del diagrama de distinta manera según el tipo de layout asociado al diagrama.				
Trazabilidad	RU-C04, RU-C10				

Tabla 31: RS-F009 – El sistema debe diferenciar los layouts

ID: RS-F010 – Acción de layout en modo nativo y transformado					
Fecha	26/02/2015	**Tipo**	Funcional		
Fuente	The Reuse Company	**Estado**	Propuesto		Cerrado
Necesidad	Esencial	Deseable		Opcional	
Prioridad	Alta	Media		Baja	
Complejidad	Alta	Media		Baja	
Objetivo	El usuario podrá realizar la acción de layout tanto en el modo nativo como en el modo transformado.				
Precondiciones	El diagrama tenga asociado un layout.				
Descripción	El sistema debe realizar una recolocación lógica tanto en el modo nativo como en el modo transformado.				
Trazabilidad	RU-C04, RU-C10				

Tabla 32: RS-F010 – Acción de layout en modo nativo y transformado

ID: RS-F011 – Cambios en la disposición de los elementos del diagrama					
Fecha	26/02/2015	**Tipo**	Funcional		
Fuente	The Reuse Company	**Estado**	Propuesto		Cerrado
Necesidad	Esencial	Deseable		Opcional	
Prioridad	Alta	Media		Baja	
Complejidad	Alta	Media		Baja	
Objetivo	El usuario podrá realizar cambios personales sobre la disposición de los elementos del diagrama, y estos se respetarán del modo nativo al transformado y viceversa.				
Precondiciones	-				

Descripción	El sistema debe preservar los cambios personales que el usuario realiza sobre el diagrama en el cambio de modo nativo a transformado y viceversa, a menos que el usuario seleccione la opción de layout para su diagrama que eliminará estos cambios realizando la recolocación lógica del layout asociado al diagrama.
Trazabilidad	RU-C04, RU-C14

Tabla 33: RS-F011 – Cambios en la disposición de los elementos del diagrama

ID: RS-F012 – Restricciones en la edición de layout de un diagrama				
Fecha	26/02/2015	**Tipo**	Funcional	
Fuente	The Reuse Company	**Estado**	Propuesto	Cerrado
Necesidad	Esencial	Deseable	Opcional	
Prioridad	Alta	Media	Baja	
Complejidad	Alta	Media	Baja	
Objetivo	El usuario no podrá cambiar el tipo de layout de su diagrama más que en el modo nativo, nunca en el transformado			
Precondiciones	-			
Descripción	El sistema debe limitar la edición de la filosofía de dibujo de un diagrama para que sólo cambien en la ventana de modo nativo.			
Trazabilidad	RU-C10			

Tabla 34: RS-F012 – Restricciones en la edición de layout de un diagrama

ID: RS-F013 – Rotación y traslación de las imágenes tras modificaciones del diagrama				
Fecha	26/02/2015	**Tipo**	Funcional	
Fuente	The Reuse Company	**Estado**	Propuesto	Cerrado
Necesidad	Esencial	Deseable	Opcional	
Prioridad	Alta	Media	Baja	
Complejidad	Alta	Media	Baja	
Objetivo	El usuario podrá modificar la situación de los elementos dentro del diagrama, en el modo nativo o transformado, implicando el giro de las imágenes en los diagramas con filosofías de dibujo que se basen en conexión.			
Precondiciones	El usuario debe haber añadido al menos un artefacto al diagrama. Debe existir al menos un tipo de artefacto con representación en la base de datos. Un artefacto debe haberse editado seleccionando un tipo de artefacto con representación. El diagrama debe tener asociado un layout que permita representaciones en los nodos.			
Descripción	El sistema debe rotar y trasladar las representaciones de los artefactos dentro del diagrama que ha modificado el usuario, ajustando la imagen a la nueva orientación de las conexiones del nodo. Esto sólo se apreciará en el modo transformado.			
Trazabilidad	RU-C13, RU-C14			

Tabla 35: RS-F013 – Rotación y traslación de las imágenes tras modificaciones del diagrama

ID: RS-F014 – Layout de circuito eléctrico basado en una filosofía serie-paralela					
Fecha	26/02/2015	**Tipo**	Funcional		
Fuente	The Reuse Company	**Estado**	Propuesto		Cerrado
Necesidad	Esencial	Deseable		Opcional	
Prioridad	Alta	Media		Baja	
Complejidad	Alta	Media		Baja	
Objetivo	El usuario podrá seleccionar para un diagrama una filosofía de dibujo de circuito eléctrico con los elementos del diagrama dispuestos en serie y en paralelo.				
Precondiciones	El usuario debe haber añadido al menos un artefacto al diagrama. Un artefacto debe haberse editado seleccionando un tipo de artefacto con representación. El diagrama debe tener asociado un layout que permita representaciones en los nodos, en concreto el de circuito eléctrico.				
Descripción	El sistema debe ofrecer al usuario la opción de aplicar un layout a su diagrama en una disposición serie-paralela para todos los elementos que estén contenidos en el mismo.				
Trazabilidad	RU-C04, RU-C12				

Tabla 36: RS-F014 – Layout de circuito eléctrico basado en una filosofía serie-paralela

4.3.1.2 Requisitos de interfaz

ID: RS-In001 – Opción de añadir nuevo tipo de artefacto					
Fecha	26/02/2015	**Tipo**	Interfaz		
Fuente	The Reuse Company	**Estado**	Propuesto		Cerrado
Necesidad	Esencial	Deseable		Opcional	
Prioridad	Alta	Media		Baja	
Complejidad	Alta	Media		Baja	
Objetivo	La barra de herramientas debe contar con una funcionalidad que permita añadir nuevos tipos de artefactos.				
Precondiciones	-				
Descripción	La barra de herramientas deberá añadir entre las opciones con las que ya cuenta con una nueva, que permita desplegar al usuario un panel en el que pueda añadir nuevos tipos de artefactos.				
Trazabilidad	RU-C04, RU-C13				

Tabla 37: RS-In001: Opción de añadir nuevo tipo de artefacto

ID: RS-In002 – Opción de layout					
Fecha	26/02/2015	**Tipo**	Interfaz		
Fuente	The Reuse Company	**Estado**	Propuesto		Cerrado
Necesidad	Esencial	Deseable		Opcional	
Prioridad	Alta	Media		Baja	

Complejidad	Alta	Media	Baja
Objetivo	La barra de herramientas debe contar con una acción para aplicar la filosofía de dibujo asignada al diagrama.		
Precondiciones	-		
Descripción	La barra de herramientas deberá contar con una opción de cargar la acción de layout.		
Trazabilidad	RU-C04, RU-C10, RU-C12		

Tabla 38: RS-In002 - Acción de layout

ID: RS-In003 – Opción de personalizar el layout del diagrama				
Fecha	26/02/2015	**Tipo**	Interfaz	
Fuente	The Reuse Company	**Estado**	Propuesto	Cerrado
Necesidad	Esencial	Deseable	Opcional	
Prioridad	Alta	Media	Baja	
Complejidad	Alta	Media	Baja	
Objetivo	La barra de herramientas debe contar con una opción para definir cuál es el layout entre los que están disponibles.			
Precondiciones	-			
Descripción	La barra de herramientas deberá contar con una opción añadir un nuevo layout al diagrama de tal manera que cambie su filosofía de dibujo.			
Trazabilidad	RU-C04, RU-C11			

Tabla 39: RS-In003 – Opción de personalizar el layout del diagrama

4.3.2 Análisis de los requisitos

En este punto se realizará un análisis de consistencia de los requisitos definidos para el proyecto mediante una matriz de trazabilidad. Gracias a esta matriz de trazabilidad se puede demostrar que el sistema que se está contemplando es consistente, y que se están cumpliendo las necesidades del cliente con los requisitos que se han especificado en el punto anterior. También tendremos en otra matriz la relación entre los casos de uso y los requisitos de sistema para mostrar la trazabilidad entre requisitos en el punto 4.3.3 Matriz de trazabilidad de Requisitos de Sistema – Casos de uso.

A continuación se muestra esta matriz de trazabilidad:

IDs	**RU-C01**	**RU-C02**	**RU-C03**	**RU-C04**	**RU-C05**	**RU-C06**	**RU-C07**	**RU-C08**	**RU-C09**	**RU-C10**	**RU-C11**	**RU-C12**	**RU-C13**	**RU-C14**	**RU-R01**	**RU-R02**	**RU-I01**	**RU-I02**
RS-F001	√	√											√			√		

RS-F002	√	√											√					
RS-F003	√	√	√			√							√			√		
RS-F004	√	√			√								√					√
RS-F005										√		√	√					
RS-F006				√	√						√		√					
RS-F007	√	√	√	√							√							
RS-F008											√	√	√					
RS-F009				√						√								
RS-F010				√						√								
RS-F011				√										√				
RS-F012										√								
RS-F013													√	√				
RS-F014				√								√						
RS-In001				√									√					
RS-In002				√						√		√						
RS-In003				√							√							

Tabla 40: Matriz de trazabilidad entre los requisitos de usuario y de sistema

4.3.3 Especificación de casos de uso

En este punto se van a detallar los casos de uso que surgen de la especificación de los requisitos de usuario.

Estos se han definido después de especificar los requisitos de sistema porque estos requisitos son previos al diseño de los casos de uso, debido principalmente a que el cliente tenía muy claros los mismos y tan sólo con algunas reuniones quedaron correctamente definidos. Aunque pueda resultar extraño debido a que habitualmente la especificación y diseño de los

casos de uso es anterior a definir los requisitos de sistema (al basarse estos normalmente en los mismos casos de uso), la especificación de casos de uso y de requisitos está íntimamente relacionada, no importando en este caso que los casos de uso se definan después de los requisitos del sistema.
Este sería otro punto que podría considerarse atípico en la redacción de la documentación de este trabajo de fin de grado, pero que se justifica en que todo este proyecto no deja de ser una mejora de uno existente, lo que plantea problemas que no se contemplarían en proyectos que son concebidos desde cero.
A continuación se muestra la plantilla que se usará para los diferentes casos de uso que se definen en este punto:

CU-###: Nombre del caso de uso	
Descripción	Descripción del caso de uso y objetivo del mismo.
Precondiciones	Condiciones que se deben cumplir antes de que el caso de uso pueda llevarse a cabo. Pueden no darse precondiciones.
Escenario	Secuencia de acciones que se ejecutarán por parte del usuario para completar el caso de uso.
Postcondiciones	Condiciones que aparecen cuando el caso de uso una vez ha sido llevado a cabo. Pueden no darse postcondiciones.
Condiciones de fallo	Condiciones que afectan al escenario y las respuestas del sistema ante estas situaciones.
Requisitos relacionados	Requisitos que están relacionados con este caso de uso.

Tabla 41: Plantilla de ejemplo para los casos de uso

El título de la tabla corresponde a un identificador único para cada caso de uso, usando la siguiente nomenclatura: **CU-###: Nombre del caso de uso.** Donde CU hace referencia a la abreviatura de caso de uso y las almohadillas corresponden al número identificador del caso de uso.

Los casos de uso que se definirán a continuación se repartirán entre los diferentes subsistemas de análisis que se contemplan en esta mejora, pero sólo incluyendo los subsistemas que requieren la interacción con el usuario. En estos casos de uso no se definirán todas las acciones posibles sobre el subsistema ya que sólo se recogerán las que hagan referencia a las mejoras desarrolladas para este proyecto de fin de grado. Así definiremos los siguientes grupos de casos de uso:

- Casos de uso relacionados con la barra de herramientas.
- Casos de uso relacionados con la interacción del usuario con los elementos del canvas.

A continuación se recogerán los diferentes casos de uso agrupados por los subsistemas de análisis, recogiéndolos además en un diagrama.

4.3.3.1 Casos de uso relacionados con la barra de herramientas

En la siguiente imagen se muestran todos los casos de uso para las mejoras desarrolladas para el subsistema de la barra de herramientas:

Una vez definido el diagrama se recogerán los diferentes casos de uso para este subsistema.

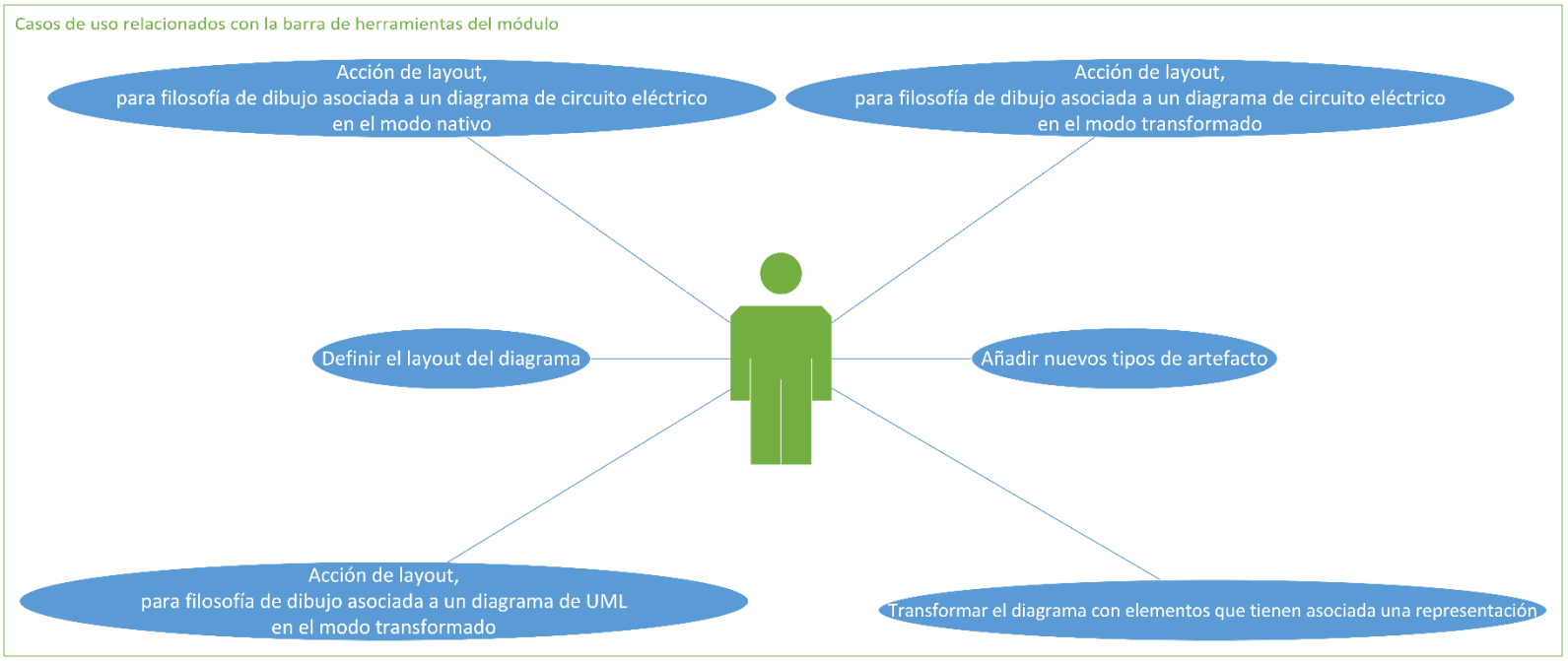

Ilustración 16: Casos de uso relacionados con la barra de herramientas

CU-001: Efectuar acción de layout, para filosofía de dibujo asociada a un diagrama de circuito eléctrico en el modo nativo	
Descripción	El usuario dispone de la opción de aplicar un layout a los elementos del diagrama. Como mejora se permite, junto con las filosofías de dibujo que ya existían en el sistema, aplicar layout de circuito eléctrico en el modo nativo
Precondiciones	Tener abierta una interfaz. El diagrama debe tener asociada una filosofía de dibujo de las que ofrece el módulo. El diagrama tiene que contar con elementos ya añadidos por el usuario.
Escenario	1. Desplegar la barra de herramientas clicando sobre ella. 2. Pulsar el icono de aplicar layout.
Postcondiciones	Los elementos que conforman el diagrama se recolocarán siguiendo la filosofía que se define en ese layout.
Condiciones de fallo	-
Requisitos relacionados	RS-F010, RS-F014, RS-In002

Tabla 42: CU-001: Efectuar acción de layout

CU-002: Efectuar acción de layout, para filosofía de dibujo asociada a un diagrama de circuito eléctrico en el modo transformado	
Descripción	El usuario dispone de la opción de aplicar un layout a los elementos del diagrama. Como mejora se permite, junto con las filosofías de dibujo que ya existían en el sistema, aplicar layout de circuito eléctrico en el modo transformado.
Precondiciones	Tener abierta una interfaz. El diagrama debe tener asociada una filosofía de dibujo de las que ofrece el módulo. El diagrama tiene que contar con elementos ya añadidos por el usuario.
Escenario	1. Desplegar la barra de herramientas clicando sobre ella. 2. Pulsar el icono de transformación. 3. En la nueva ventana de transformación, desplegar de nuevo la barra de herramientas clicando sobre ella. 4. Pulsar el icono de aplicar layout.
Postcondiciones	Los elementos que conforman el diagrama se recolocarán siguiendo la filosofía que se define en ese layout también en la ventana de transformación
Condiciones de fallo	-
Requisitos relacionados	RS-F010, RS-F014, RS-In002

Tabla 43: CU-002: Efectuar acción de layout, para filosofía de dibujo asociada a un diagrama de circuito eléctrico en el modo transformado

CU-003: Efectuar acción de layout, para filosofía de dibujo asociada a un diagrama UML en el modo transformado	
Descripción	El usuario dispone de la opción de aplicar un layout a los elementos del diagrama. Como mejora se permite, junto con las filosofías de dibujo que ya existían en el sistema, aplicar layout de UML en el modo transformado.
Precondiciones	Tener abierta una interfaz. El diagrama debe tener asociada una filosofía de dibujo de las que ofrece el módulo. El diagrama tiene que contar con elementos ya añadidos por el usuario.
Escenario	1. Desplegar la barra de herramientas clicando sobre ella. 2. Pulsar el icono de transformación. 3. En la nueva ventana de transformación, desplegar de nuevo la barra de herramientas clicando sobre ella. 4. Pulsar el icono de aplicar layout.
Postcondiciones	Los elementos que conforman el diagrama se recolocarán siguiendo la filosofía que se define en ese layout.
Condiciones de fallo	-
Requisitos relacionados	RS-F010, RS-In002

Tabla 44: CU-003: Efectuar acción de layout, para filosofía de dibujo asociada a un diagrama UML en el modo transformado

CU-004: Añadir nuevos tipos de artefactos	
Descripción	El usuario dispone de la opción de crear nuevos tipos de artefactos desde el módulo sin necesidad de volver al Knowledge Manager.
Precondiciones	Tener abierta una interfaz.
Escenario	1. Desplegar la barra de herramientas clicando sobre ella. 2. Pulsar el icono de añadir nuevo tipo de artefacto. 3. Rellenar los campos con la información del nuevo tipo de artefacto: - Nombre: Identifica el nuevo tipo de artefacto. - Ángulo: Indica la orientación de la representación gráfica del nuevo tipo de artefacto si este tiene asociada alguna. Puede seleccionarse la opción no rotar en caso de no necesitar el sentido de la imagen. - Icono: En este campo se asignará una imagen que sirva de icono para este nuevo tipo de artefacto. No es obligatorio crear un nuevo tipo de artefacto con representación gráfica. - Tipo de representación: Genérica sino se está añadiendo un tipo de artefacto que sea un diagrama con una filosofía de dibujo asociada, en caso contrario, se selecciona la filosofía de dibujo que se quiera asignar a ese tipo de artefacto que se está creando.

	4. Clicar en el botón de aceptar para añadir el nuevo tipo de artefacto, o en cancelar para cancelar la operación de guardado.
Postcondiciones	Se añadirá un nuevo tipo de artefacto a la base de datos que podrá usarse en cualquier artefacto que se añada o modifique, o se podrá asignar este nuevo tipo de artefacto a un diagrama si cuenta con una filosofía de dibujo asociada.
Condiciones de fallo	-
Requisitos relacionados	RS-F001, RS-F004, RS-F006, RS-In001

Tabla 45: CU-004: Añadir nuevos tipos de artefactos

CU-005: Definir layout del diagrama	
Descripción	El usuario dispone de la opción de elegir el layout asignado a su diagrama a través de una selección del tipo de artefacto, de tal manera que pueda elegir uno basado en las filosofías de dibujo que ofrece el módulo al usuario.
Precondiciones	Tener abierta una interfaz. Contar con al menos un tipo de artefacto que cuente con una filosofía de dibujo asociada serializado.
Escenario	1. Desplegar la barra de herramientas clicando sobre ella. 2. Pulsar el icono de cambiar el tipo de diagrama a transformar 3. Seleccionar el tipo de artefacto con una filosofía de dibujo asociada entre los que se ofrecen. 4. Clicar en el botón de aceptar para guardar el nuevo tipo de artefacto con el que se asocia el diagrama para tener una nueva filosofía de dibujo, o en cancelar para cancelar la operación de guardado.
Postcondiciones	Se asignara al diagrama el tipo de artefacto seleccionado, permitiendo que el usuario aplique con la opción de layout esta nueva filosofía de dibujo, tanto en el modo nativo como en el transformado. También se mostrará el cambio de tipo de artefacto asignado al diagrama en la etiqueta inferior a la barra de herramientas que muestra el nombre del tipo de artefacto que tiene asignado actualmente el diagrama.
Condiciones de fallo	-
Requisitos relacionados	RS-F007, RS-F012, RS-In003

Tabla 46: CU-005: Definir layout del diagrama

CU-006: Transformar el diagrama con elementos que tienen asociada una representación	
Descripción	El usuario dispone de la opción de transformar un diagrama nativo a un diagrama de transformación pulsando el icono de Pasar a modo de transformación y viceversa. Como mejora en este caso de uso recogido por mis compañeros del año pasado, si un artefacto tiene asignado un tipo de artefacto con representación, esta se mostrará como un icono en el modo transformado siempre que la filosofía de dibujo permita representaciones (Por ejemplo el diagrama de secuencia no las permite). Además teniendo en cuenta la orientación del artefacto con sus conexiones a relaciones entrantes y salientes, y si se da el caso de que la filosofía de dibujo asociada al diagrama se basa en conexión, los iconos girarán para formar un eje con respecto a los puntos por los que se unen las relaciones entrantes y salientes al nodo.
Precondiciones	Tener abierta una interfaz.
Escenario	1. Desplegar la barra de herramientas clicando sobre ella. 2. Pulsar el icono de Pasar a modo de transformación y viceversa. 3. Clicar en el botón de cerrar en la nueva ventana que muestra el diagrama transformado.
Postcondiciones	El diagrama volverá a mostrarse en su modo nativo cerrándose la nueva ventana del modo transformado, si se ha realizado algún cambio en el modo transformado (posición de los nodos por ejemplo) esté se verá reflejado también en el modo nativo. Estos últimos cambios se detallan en profundidad en el CU-005.
Condiciones de fallo	-
Requisitos relacionados	RS-F003, RS-F005, RS-F010, RS-F011, RS-F013

Tabla 47: CU-006: Transformar el diagrama con elementos que tienen asociada una representación

4.3.2.2 Casos de uso relacionados con la interacción del usuario con los elementos del canvas

En la siguiente imagen se muestran todos los casos de uso para las mejoras desarrolladas para el subsistema de la interacción del usuario con los elementos del canvas:

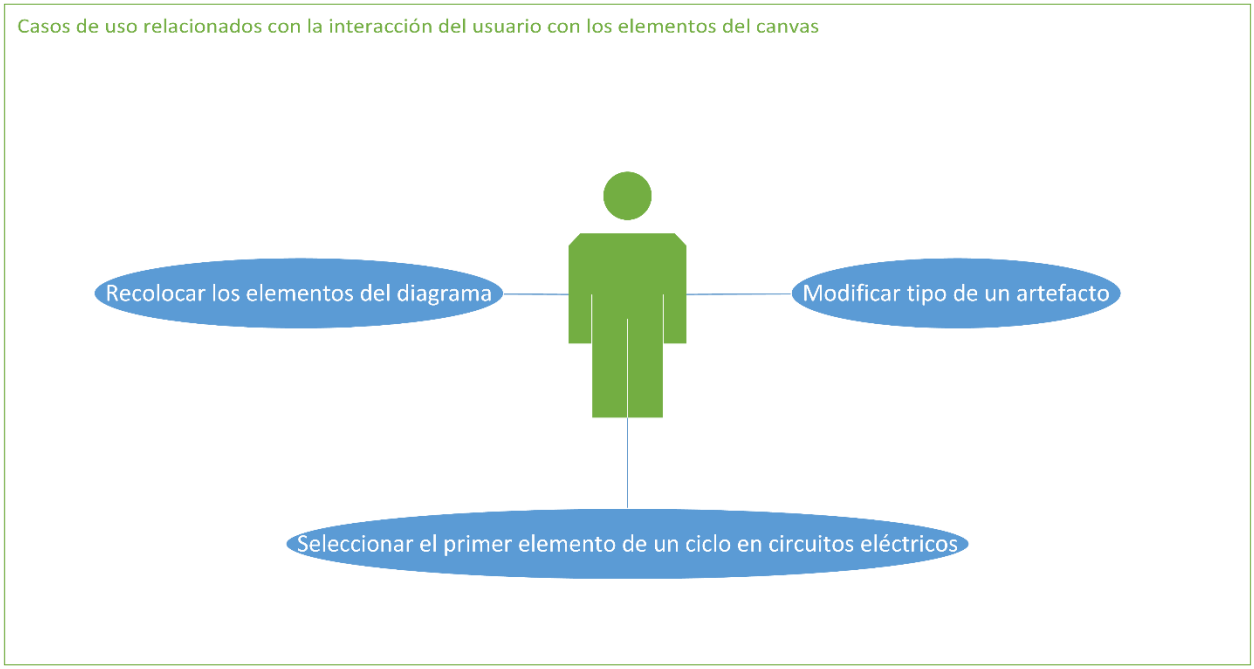

Ilustración 17: Casos de uso relacionados con la interacción del usuario con los elementos del canvas

A continuación se recogen todos los casos de uso relacionados con este subsistema:

CU-007: Recolocar los elementos del diagrama	
Descripción	El usuario dispone de la opción de si se ha realizado alguna modificación sobre el diagrama en el modo transformado, ya sea en la posición que ocupan los nodos, o incluso las relaciones, estos cambios se podrán reflejar en el modo nativo cuando se cierre la ventana transformada. De igual manera, si el usuario decide aplicar la acción de layout en la ventana transformada, estos cambios se reflejarán en el modo nativo del diagrama. Del mismo modo sucede si el usuario realiza cambios en el modo nativo, reflejándose estos en el modo transformado del diagrama. Esta recolocación no afecta sólo a la posición de los nodos o las relaciones, sino que puede afectar a la rotación de la representación de un nodo del diagrama en caso de que esté cuente con una filosofía de dibujo basada en conexión.
Precondiciones	Tener abierta una interfaz.
Escenario	1. Desplegar la barra de herramientas clicando sobre ella. 2. Pulsar el icono de Pasar a modo de transformación y viceversa. 3. Realizar la acción de layout sobre el diagrama en la ventana de transformación, o recolocar nodos y relaciones.

	4. Clicar en el botón de cerrar en la nueva ventana que muestra el diagrama transformado. 5. Repetir procedimiento si fuese necesario modificando las posiciones de nodos y relaciones, o aplicando la acción de layout en el modo nativo.
Postcondiciones	Si se ha realizado alguna modificación sobre el diagrama en el modo transformado, ya sea en la posición que ocupan los nodos, o incluso las relaciones, estos cambios se verán reflejados en el modo nativo cuando se cierre la ventana transformada. De igual manera, si el usuario decide aplicar la acción de layout en la ventana transformada, estos cambios se reflejarán en el modo nativo del diagrama.
Condiciones de fallo	Si el usuario en vez de cerrar la ventana de transformación en el botón de cerrar, cierra la ventana directamente, no guardará los cambios y se mantendrá el diagrama en su modo nativo sin ningún cambio.
Requisitos relacionados	RS-F005, RS-F010, RS-F011, RS-F013

Tabla 48: CU-007: Recolocar los elementos del diagrama

CU-008: Modificar tipo de un artefacto.	
Descripción	El usuario dispone de la opción de modificar un artefacto que se encuentre en el canvas, de tal manera que pueda modificar el tipo de artefacto que tiene asignado, asignándole por ejemplo un tipo de artefacto que cuente con representación. Este caso de uso se va a centrar sólo en este aspecto.
Precondiciones	Tener abierta una interfaz. Debe existir algún artefacto en el diagrama en su modo nativo
Escenario	1. Doble clic sobre el artefacto. 2. Se despliega una ventana con la información que contiene el artefacto. 3. El usuario modifica el tipo de artefacto. 4. El usuario pulsa el botón aceptar.
Postcondiciones	Las modificaciones de tipo de artefacto sobre el artefacto se han almacenado correctamente. Si se pasa al modo transformado y el artefacto tiene un tipo con representación, esta se mostrará en forma de un icono en el diagrama.
Condiciones de fallo	-
Requisitos relacionados	RS-F002

Tabla 49: CU-008: Modificar tipo de un artefacto.

CU-009: Seleccionar primer elemento de un ciclo en circuitos eléctricos.	
Descripción	El usuario dispone de la opción de modificar un nodo (ya sea un término o un artefacto) que se encuentre en el canvas, de tal manera que pueda marcarlo como el primer elemento dentro de un circuito eléctrico que forma un ciclo.
Precondiciones	Tener abierta una interfaz. Debe existir algún término u artefacto en el diagrama en su modo nativo. El diagrama debe ser de circuito eléctrico.
Escenario	1. Doble clic sobre el nodo. 2. Se despliega una ventana con la información que contiene el nodo. 3. El usuario marca la casilla de este como primer elemento del ciclo. 4. El usuario pulsa el botón aceptar. 5. Saltará un mensaje en caso de que exista otro elemento marcado como el primero (por defecto el sistema seleccionará el primer elemento que se haya añadido al canvas). 6. Si se acepta se cambiará el antiguo primer elemento por el que se está editando, en caso contrario se conservará el antiguo.
Postcondiciones	Las modificaciones del nodo se han almacenado correctamente.
Condiciones de fallo	-
Requisitos relacionados	RS-F014

Tabla 50: CU-009: Seleccionar primer elemento de un ciclo en circuitos eléctricos.

4.3.3 Matriz de trazabilidad de Requisitos de Sistema – Casos de Uso

En este punto se definirá una matriz de trazabilidad para establecer la relación entre los requisitos del sistema y los casos de uso:

IDs	**CU-01**	**CU-02**	**CU-03**	**CU-04**	**CU-05**	**CU-06**	**CU-07**	**CU-08**	**CU-09**
RS-F001				√					
RS-F002								√	
RS-F003						√			
RS-F004				√					
RS-F005						√	√		
RS-F006				√					
RS-F007					√				
RS-F008									
RS-F009									
RS-F010	√	√	√			√	√		

RS-F011						√	√		
RS-F012					√				
RS-F013						√	√		
RS-F014	√	√							√
RS-In001				√					
RS-In002	√	√	√						
RS-In003					√				

Tabla 51: Matriz de trazabilidad de Requisitos de Sistema – Casos de Uso

A continuación se había pensado incluir el análisis de casos de uso, pero se ha decidido incluir este análisis en el apartado de diseño, ya que en esta parte de análisis se considera que se perderá mucha información debido a que los diagramas de secuencia que se incluirían en este punto no son exactos, sino que tienen un borrador de cómo sería este caso de uso, mientras que en el Análisis de casos de uso reales, se recoge el diseño real de los mismos.

Así mismo el análisis de clases es bastante limitado en este proyecto, ya que en su gran mayoría las clases que se deben implementar para cubrir las funcionalidades de las nuevas mejoras ya existen. Es por tanto más conveniente mostrar todas las mejoras, así como diagramas de secuencia entre las mismas en el apartado de Diseño de clases. Sin embargo se recogerán las nuevas clases que se han concebido como forma de cubrir las necesidades del cliente en el siguiente apartado, así como las mejoras sobre las clases existentes.

4.4 Análisis de clases

Teniendo en cuenta las necesidades de este proyecto, y sabiendo que consiste en una mejora, las nuevas clases que se han concebido para cumplir las necesidades del cliente son pocas, ya que en su mayoría, las clases existentes son mejorables sin necesidad de crear clases nuevas.

Por ello se ha decidido recoger en este apartado las principales mejoras que se implementarán para cubrir las necesidades del cliente en esta etapa de desarrollo del sistema, junto a las nuevas clases que se han creado.

Así distinguiremos los siguientes tipos de clases según el modelo que se estableció en la etapa anterior de desarrollo:

- **Clase de entidad:** Son clases empleadas para almacenar información de las mismas, sirviendo de modelo.
- **Clase de interfaz de usuario:** Son clases que se encargan de proporcionar los medios para que el usuario pueda interactuar con la aplicación. Para ello se usan formularios, ventanas, mensajes efímeros, etc.
- **Clases de control:** Estas clases se encargan de gestionar toda la lógica de negocio del sistema.

4.4.1 Identificación de responsabilidades y atributos

A continuación describiremos cada una de las clases describiendo sus atributos, responsabilidades y operaciones. Todas estas clases en su mayoría ya están concebidas desde la etapa anterior, por lo no muestran más que las nuevas funcionalidades que poseen para cubrir las nuevas necesidades del cliente.

El formato que se usará para describir las clases se muestra en la siguiente tabla que se usará como plantilla:

CL-##: Nombre de la clase	
Responsabilidades	Se describirá cuál es su función dentro del sistema
Atributos	Atributos con los que cuenta para albergar datos
Funcionalidades	Conjunto de funciones que implementa

Tabla 52: Plantilla usada para la descripción de las clases

A continuación se describirán todas las nuevas clases agrupada según la clasificación realizada anteriormente, además dentro de cada clasificación se mostrarán las clases que

son nuevas y las que existen pero contienen mejoras, ya que en este último caso sólo se mostrarán los atributos y funcionalidades nuevas añadidos y su función.

4.4.1.1 Clases de entidad

En este punto se recogerán las clases que funcionarán como objetos para el modelo de datos de este sistema.

- **Clases nuevas:**

CL-01: Propiedades del circuito	
Responsabilidades	Se encarga de interactuar con la capa de datos (modelo), para recoger los datos relacionados con las propiedades de un elemento de un circuito eléctrico, permitiendo aplicar la implementación de los algoritmos presentes en el apartado de Proceso de selección de algoritmos para dibujar diagramas de circuitos eléctricos.
Atributos	- Dos estructuras de datos para almacenar las relaciones entrantes y salientes de un nodo de un circuito eléctrico. - Dos atributos para almacenar el identificador del primer y último nodo de un ciclo, si el nodo forma parte de uno.
Funcionalidades	-

Tabla 53: CL-01: Propiedades del circuito

CL-02: Propiedades generales del nodo	
Responsabilidades	Se encarga de interactuar con la capa de datos (modelo), para recoger los datos relacionados con las representaciones que se cargan en los nodos del diagrama, mostrando estas imágenes en la transformación del diagrama.
Atributos	- Un string en Base 64 para almacenar la imagen. - La altura y la anchura de la imagen.
Funcionalidades	-

Tabla 54: CL-02: Propiedades generales del nodo

CL-03: Tipo de representación	
Responsabilidades	Se encarga de interactuar con la capa de datos (modelo), para recoger los datos relacionados con las representaciones que se cargan en los tipos de artefactos nuevos asociados a nodos del diagrama que crea el usuario a través de los formularios nuevos.
Atributos	- Un string en Base 64 para almacenar la imagen. - La imagen que carga el usuario, con su orientación, el nombre del tipo de artefacto y el tipo de representación si la tiene.
Funcionalidades	Métodos para convertir la imagen desde el formato original al formato especificado para la base de datos. También desde el formato de la base de datos a string en Base 64 que se pueda mandar a JavaScript mostrándose en la interfaz. Funciones para rotar las imágenes.

Tabla 55: CL-03: Tipo de representación

- **Mejoras sobre clase existentes:**

CL-04: Graph	
Responsabilidades	Se encarga de interactuar con la capa de datos (modelo), para recoger los datos relacionados con el gráfico que se representa.
Atributos	- Tipo de representación asociada al diagrama - Atributos que indican el tipo de circuito eléctrico, entre los casos definidos en el apartado de Configuraciones de circuitos eléctricos según las necesidades del cliente. - Atributos necesarios para recoger los datos de las componentes fuertemente conexas que necesita el algoritmo de Tarjan en su implementación.
Funcionalidades	-

Tabla 56: CL-04: Gráfico

CL-05: Relation	
Responsabilidades	Se encarga de interactuar con la capa de datos (modelo), para recoger los datos relacionados con las relaciones del diagrama.
Atributos	- Dos atributos que almacene la posición de los puntos de la relación (los puntos que se unen y la forman en JavaScript). - Un booleano para comprobar si la relación forma parte de un ciclo para circuitos eléctricos. - Dos booleanos para saber si la relación es la primera o la última evaluada en el ciclo.
Funcionalidades	-

Tabla 57: CL-05: Relación

CL-06: Node	
Responsabilidades	Se encarga de interactuar con la capa de datos (modelo), para recoger los datos relacionados con los nodos del diagrama.
Atributos	- Dos booleanos para saber si el nodo es el primero o el último, usados en la configuración de circuitos eléctricos. - Un booleano para saber si el nodo ha sido visitado cuando se aplica el algoritmo de Tarjan. - La posición que ocupa el nodo en el diagrama.
Funcionalidades	-

Tabla 58: CL-06: Nodo

4.1.1.2 Clases de interfaz

En este apartado se recogerán las nuevas clases, así como las existentes y modificadas que sirven al usuario para interactuar con el sistema en forma de diálogos, ventanas, formularios…

- **Clases nuevas:**

CL-07: Formulario para crear nuevo tipo de artefacto	
Responsabilidades	Interfaz para recoger los campos que son necesarios para crear un nuevo tipo de artefacto.
Atributos	- Objeto que contiene el nuevo tipo de artefacto
Funcionalidades	Permite al usuario añadir: - Nombre del tipo de artefacto. - Representación en forma de imagen con visor de la misma (Se representa un pequeño icono con la imagen). - Orientación de la imagen. - Tipo de representación asociada al nuevo tipo de artefacto (si es un diagrama). - Guardar en la base de datos el nuevo tipo de artefacto que se ha creado.

Tabla 59: CL-07: Formulario para crear nuevo tipo de artefacto

- **Mejoras sobre clases existentes:**

CL-08: Formulario para editar artefacto del diagrama	
Responsabilidades	Interfaz para recoger atributos que un usuario modifica de un artefacto.
Atributos	-

Funcionalidades	Permite al usuario editar, cuando el diagrama es de circuito eléctrico, un artefacto para marcarlo como primero en caso de existir un ciclo.

Tabla 60: CL-08: Formulario para editar artefacto del diagrama

CL-09: Formulario para editar término del diagrama	
Responsabilidades	Interfaz para recoger atributos que un usuario modifica de un término.
Atributos	-
Funcionalidades	Permite al usuario editar, cuando el diagrama es de circuito eléctrico, un término para marcarlo como primero en caso de existir un ciclo.

Tabla 61: CL-09: Formulario para editar término del diagrama

CL-10: Formulario para editar el tipo de artefacto asignado al diagrama	
Responsabilidades	Interfaz para recoger el cambio de tipo de artefacto asociado a un diagrama.
Atributos	- Nuevo selector con todos los tipos de artefactos serializados en la base de datos del KM que tienen algún tipo de filosofía de dibujo asociada
Funcionalidades	Permite al usuario editar el tipo de artefacto asociado a un diagrama, mostrándole al usuario todos los tipos de artefacto que cuentan con una filosofía de dibujo asociada.

Tabla 62: CL-10: Formulario para editar el tipo de artefacto asignado al diagrama

Las modificaciones sobre clases de la librería gráfica se recogerán directamente en el punto de Diseño de clases. Ya que es necesario especificar el diseño de las mismas con todas sus funciones reales para llegar a comprender su funcionalidad dentro del módulo.

4.1.1.3 Clases de control

En este apartado del análisis de clases se especificarán las que están encargadas de la lógica de negocio de la aplicación. Casi todas las funcionalidades más importantes del sistema se implementarán aquí, sirviendo así de enlace entre las clases de interfaz y las de entidad. La arquitectura de este sistema viene muy bien detallada en el punto de Definición de los niveles de arquitectura.

En este punto además cabe destacar que no se ha creado ninguna clase nueva, sólo se mejorarán las siguientes clases:

CL-11: Mejoras en graphicalArtifactUserControl	
Responsabilidades	Clase que contiene la lógica de negocio asignada al módulo de gestión de diagramas, tanto en el modo nativo como en transformado.
Atributos	-
Funcionalidades	- Función para aplicar el algoritmo de Tarjan sobre el diagrama detectando ciclos para poder colocarlos correctamente, tanto en el modo nativo como cuando se carga la transformación. - Función para ordenar el diagrama desde el primer elemento seleccionado por el usuario, si el diagrama es de circuito eléctrico. - Función para recalcular las posiciones del primer y último nodo en diagramas de circuitos eléctricos como se especifica en el punto de Configuraciones de circuitos eléctricos según las necesidades del cliente. - Función para serializar en la base de datos los nuevos tipos de artefactos con todas sus propiedades desde la interfaz de nuevos tipos de artefactos, devolviendo el objeto para su serialización.

Tabla 63: CL-11: Mejoras en graphicalArtifactUserControl

CL-12: Mejoras en Transformation	
Responsabilidades	Clase que contiene la lógica de negocio asignada al módulo de transformación de diagramas.
Atributos	-
Funcionalidades	- Función para tomar las imágenes de los tipos de artefactos asociados a los nodos (artefactos) de un diagrama, rotándolas si fuera necesario según el sentido del diagrama (sólo circuitos eléctricos) y convirtiéndolas al formato adecuado para JavaScript devolviéndose como atributos de objetos, que son los nodos que se pasan a las librerías gráficas para representarlos.

Tabla 64: CL-12: Mejoras en Transformation

4.5 Definición de interfaces de usuario

En este punto, y siguiendo el modelo definido en el estándar elegido para la documentación, Métrica V3, se especificarían las interfaces entre el sistema y el usuario, incluyéndose el formato de pantalla, diálogos e incluso el formato de peticiones de otros subsistemas. Sin embargo, como este proyecto consiste en una mejora del sistema creado por mis compañeros del año pasado, se ha decidido restructurar el punto para mostrar sólo las mejoras que se han realizado sobre estas interfaces, que ayudan en gran medida a comprender la mayoría de las nuevas opciones que ofrece esta nueva etapa de desarrollo de este módulo.

Se ha decidido incluir este punto en la documentación debido a que la parte de la interfaz es una de las más importantes de este proyecto debido a las mejoras en cuanto a carga gráfica que se han desarrollado para esta mejora.

4.4.2 Especificación de principios generales de la interfaz.

En el diseño de la interfaz es fundamental recoger las necesidades, experiencia y capacidades de los usuarios. Es además muy importante conocer los futuros usuarios de la aplicación, es decir, los potenciales clientes que se pueden conseguir gracias a las mejoras desarrolladas en este trabajo de final de grado.

En este sentido este proyecto cuenta con una gran ventaja que permite definir y conocer perfectamente las necesidades del usuario final. Esto es gracias a que la definición y principios de la interfaz fueron claramente definidos en el trabajo de fin de grado desarrollado por Pablo Sánchez Pérez, y el propio cliente ha decidido que sigamos con las propiedades del diseño de la etapa anterior.
A continuación se recogerán estos principios para identificar la interfaz citando al trabajo de fin de grado de Pablo Sánchez Pérez:

Mínimas acciones para realizar los casos de uso	*Siempre que sea posible, hay que minimizar el número de interacciones necesarias que ha de efectuar el usuario para realizar un caso de uso.*
Ubicación familiar de elementos	*Ubicación familiar de elementos de la paleta de edición, barra de herramientas y canvas. La ubicación debe recordar al usuario a otras herramientas de edición similares que pueda usar o haber usado (Word, Photoshop…).*
Confirmación de acciones destructivas	*Una librería en la que su desarrolladores han hecho un estupendo trabajo. Es muy vistosa y ofrece funcionalidades para diagramar.*

Confirmación de acciones destructivas	*Si el resultado de la interacción del usuario es una acción potencialmente destructiva, se pedirá al usuario confirmación de que realmente la quiere realizar.*
Presentación de ayudas con la mínima agresión	*Las notificaciones y ayudas han de implementarse de tal modo que suponga la mínima agresión posible para la usabilidad del usuario.*
Implementar facilidades de "deshacer" y "rehacer".	*El comando deshacer retornara el sistema a un estado anterior al del error. La implementación de este comando es siempre costosa, pero es habitual en prácticamente todos los sistemas actuales.*
Los colores deberán ser homogéneos y de tono pastel.	*El cliente demanda colores pastel para la interfaz, el uso de colores muy vivos puede ser perjudicial para confortabilidad del cliente durante el uso del sistema.*
Atajos de teclado para las acciones comunes.	*Cuando un usuario se ha acostumbrado a manejar la aplicación, este demanda soltura para realizar sus acciones mas eficientemente. Para ello hay que proporcionarle atajos de teclado que agilicen las tareas comunes (ctrl + c, ctrl + v, tab, supr, ctrl + z…etc).*

Tabla 65: Principios seguidos el año pasado para definir la interfaz

4.4.3 Especificación de formatos individuales de la interfaz de pantalla

En este punto se recogerán los formatos individuales de las pantallas que se han mejorado en este proyecto.

Para ello, y aunque realmente toda la interfaz está basada en una única ventana que contiene la paleta de edición, la barra de herramientas y el canvas, sólo se han realizado cambios sobre la barra de herramientas, y sobre las ventanas que se despliegan con los diferentes formularios que se han creado para las nuevas funciones solicitadas por el cliente para esta etapa de desarrollo.

4.4.3.1 Mejoras sobre la interfaz de edición

En este punto se recogerán las mejoras relacionadas con toda la pantalla de edición.
A continuación se muestra la pantalla de edición en su configuración original y en su nuevo diseño con las mejoras añadidas (siempre con la barra de herramientas desplegada):

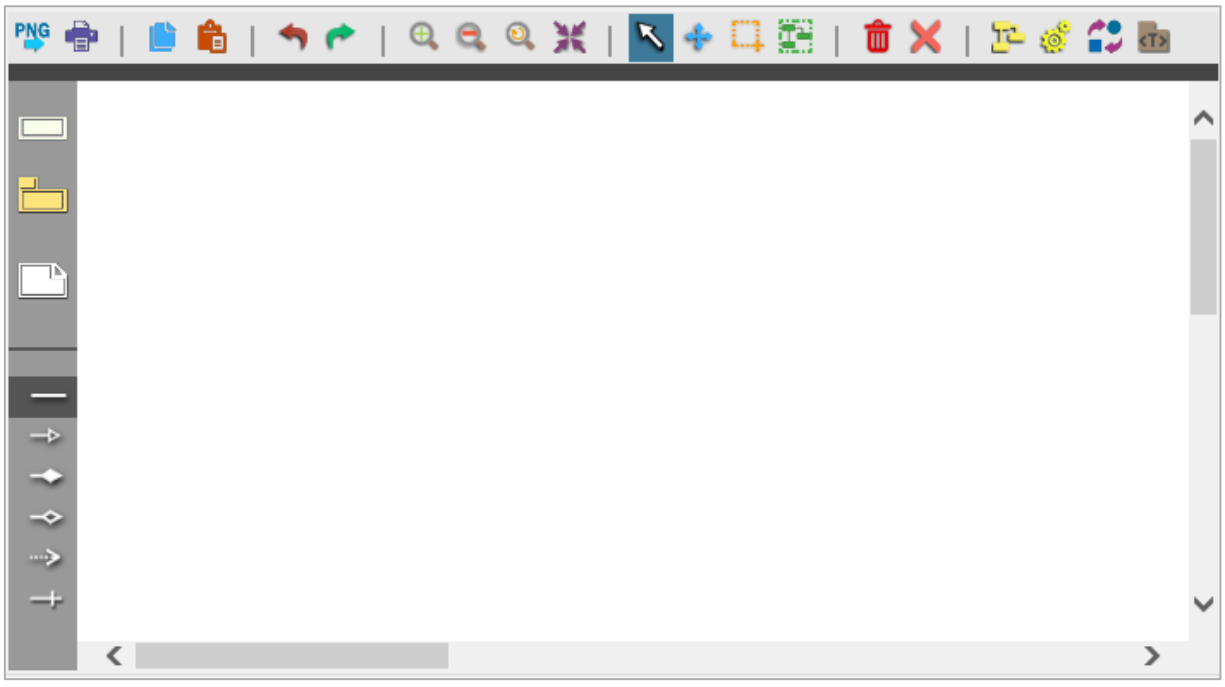

Ilustración 18: Pantalla de edición en su configuración original con la barra de herramientas desplegada.

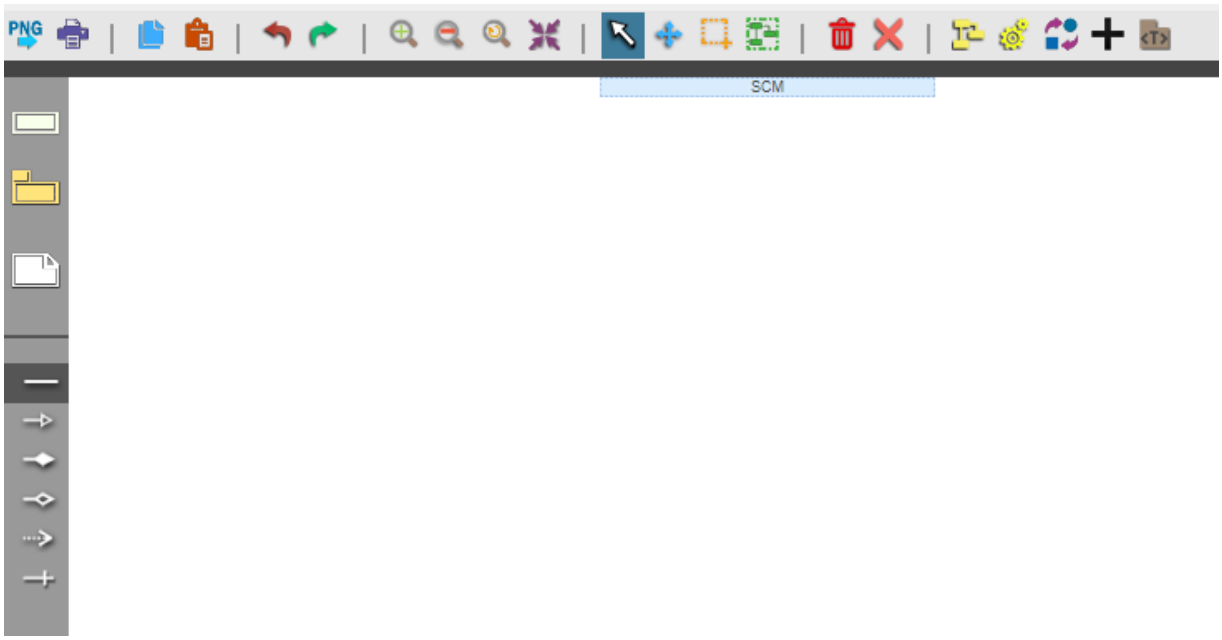

Ilustración 19: Pantalla de edición con modificaciones

Aunque en un principio no se aprecien muchas modificaciones, estás son fundamentales para entender las nuevas funcionalidades que se han desarrollado.

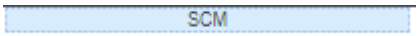

Ilustración 20: Identificador de tipo de diagrama

La primera de ellas consiste en el identificador del tipo de diagrama que sobre el que se está trabajando, y que es necesaria para distinguir el tipo de layout que se aplicará al realizar la acción de layout y que se aprecia en la figura de arriba.

La segunda consiste en el siguiente botón:

Ilustración 21: Botón de añadir nuevo tipo de artefacto

Este botón de la barra de herramientas es una de las nuevas mejoras más importantes, ya que es el botón que permite crear nuevos tipos de artefactos para dotar a los mismos con representaciones en forma de iconos. También permite definir nuevos tipos de artefactos con filosofías de dibujo asociadas que permiten crear nuevos tipos de diagrama.

Otra mejora sobre la paleta de edición sólo se puede apreciar cuando el tipo de diagrama que se ha seleccionado es de circuito eléctrico. En este caso se muestran los siguientes cambios que se explicarán a continuación de la captura de pantalla:

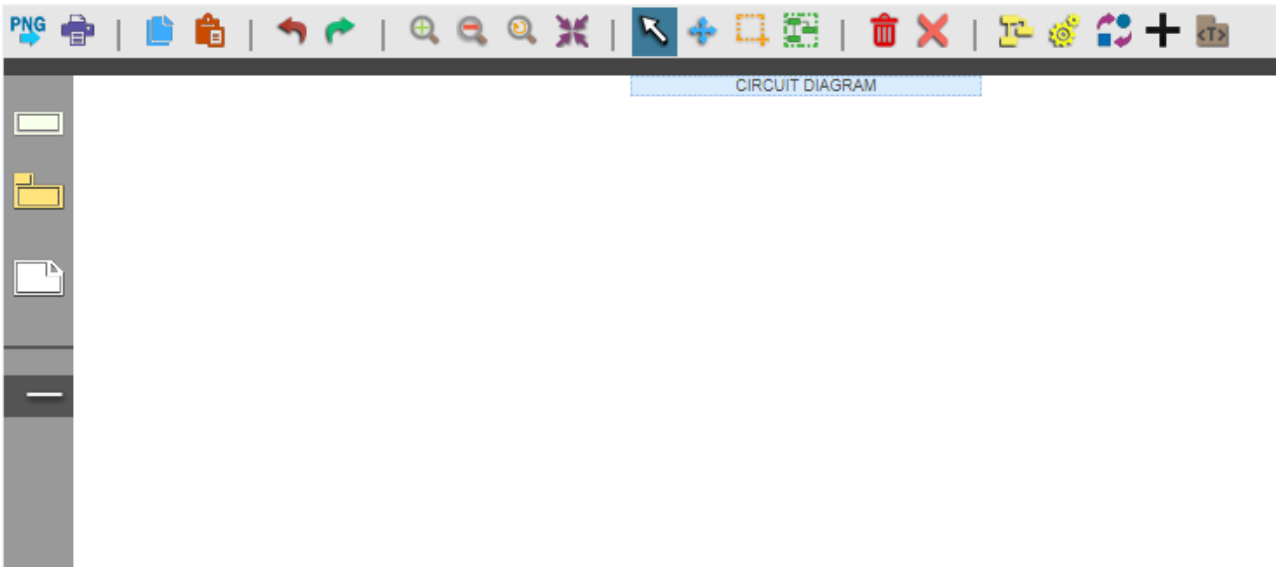

Ilustración 22: Paleta de edición cuando se edita un circuito eléctrico

Como se puede apreciar en la imagen superior, la paleta de edición limita los tipos de relaciones que se pueden añadir a diagramas de circuitos eléctricos sólo a relaciones de asociación, ya que es uno de los requerimientos del cliente al modificar las interfaces para añadir todas las nuevas funciones.

4.4.3.2 Mejoras sobre la interfaz de transformación

En este punto se mostrarán los cambios que ha sufrido la interfaz de transformación con respecto a su configuración original.

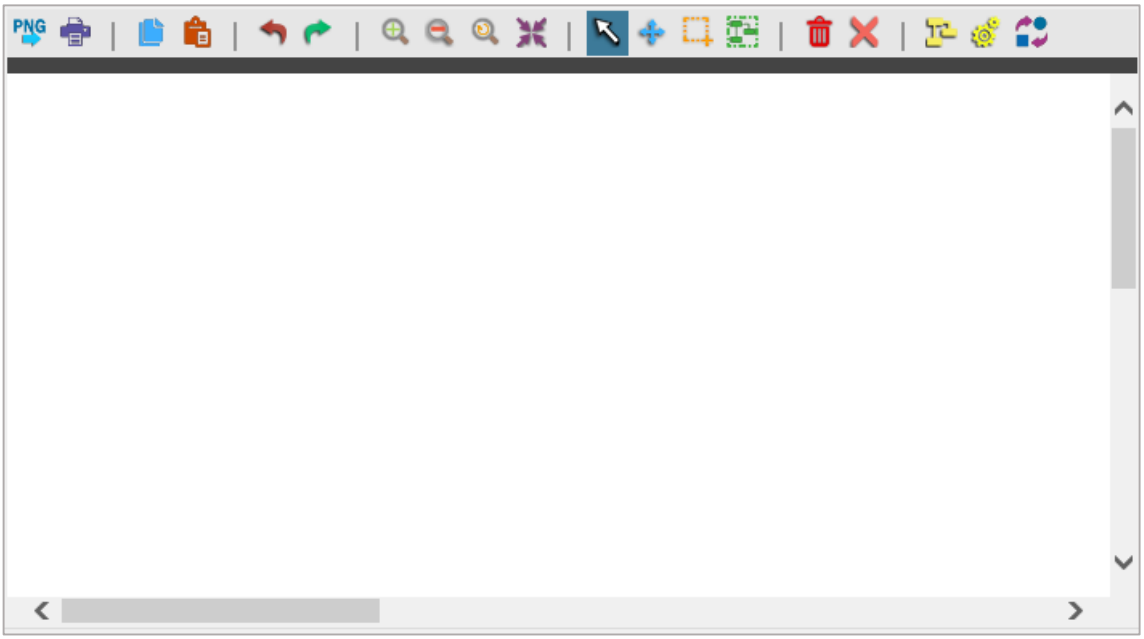

Ilustración 23: Ventana de transformación original

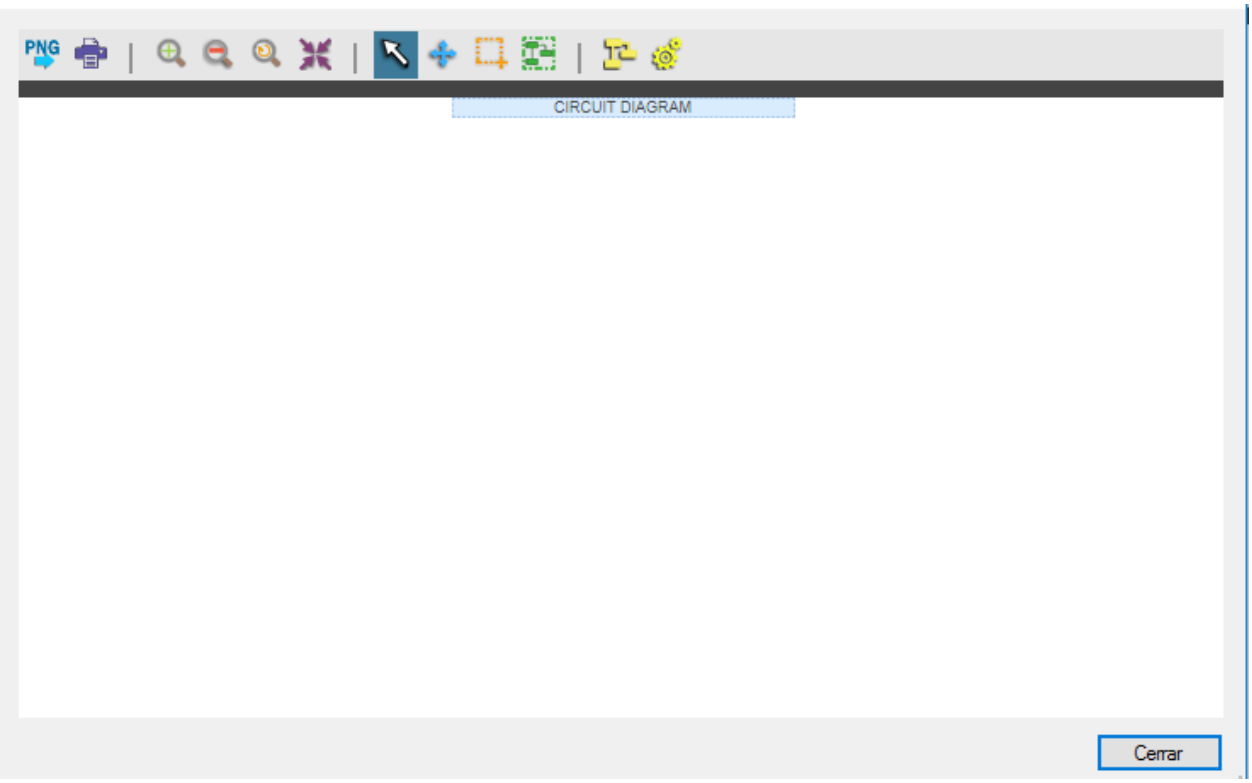

Ilustración 24: Pantalla de transformación con modificaciones

Como se puede apreciar en las imágenes anteriores, los cambios en la ventana de transformación radican en eliminar algunas de las opciones de la barra de herramientas, ya que estás realmente no tenían ninguna función en la ventana de transformación y sólo se desplegaba un mensaje efímero al cliente indicando que no se podía efectuar.

Como se aprecia en la captura, y teniendo en cuenta lo que se ha ido documentando en este proyecto, las únicas acciones que se permiten en la ventana de transformación están relacionadas con la recolocación de los elementos del diagrama en el canvas. Es por ello que se permiten las acciones de seleccionar todos los elementos o unos pocos seleccionados (mediante el recuadro de selección que el usuario aplica con un movimiento con el ratón), pero ninguna acción de eliminar elementos, cosa que en la ventana de transformación original tampoco se permitía. Además una diferencia de esta ventana mejorada con respecto a la antigua, radica en que la antigua si mostraba esas opciones para eliminar elementos, pero no podían aplicarse sobre el diagrama.

Otro punto fundamental de la mejora sobre la ventana de transformación consiste en el botón de acción de layout, ya que se permite realizar la acción de layout en la ventana de transformación, cosa que originalmente tampoco se permitía aunque en la barra de herramientas apareciese la opción.

4.4.4 Identificación de diálogos

En este punto se recogerán las diferentes mejoras que se han desarrollado con respecto a la comunicación con el usuario. Sobre todo, las diferentes mejoras que se han realizado sobre los formularios C#, y que son necesarios para todas las mejoras que se van a realizar.
Así a continuación se detallarán estás mejoras.

4.4.4.1 Formularios C#

Estos formularios surgen ante la necesidad de comunicar la interfaz con otros subsistemas atendiendo las nuevas necesidades recogidas en los casos de uso de esta mejora. Todos los formularios que se muestran a continuación son mostrados en inglés, aunque también se han añadido recursos para que se muestren en español según la región en la que se ejecute la aplicación, cumpliendo así con las necesidades del cliente.

Así y observando los casos de uso que se han definido tendremos los siguientes formularios nuevos o modificados:

- **Añadir nuevo tipo de artefacto:**
  Para este caso de uso se definirá una nueva ventana que se desplegará cuando el usuario necesite añadir nuevos tipos de artefactos para su diagrama.

  La ventana creada, según el estándar definido para todas las ventanas tanto del Knowledge Manager, como para este módulo, se muestra en la captura de a continuación:

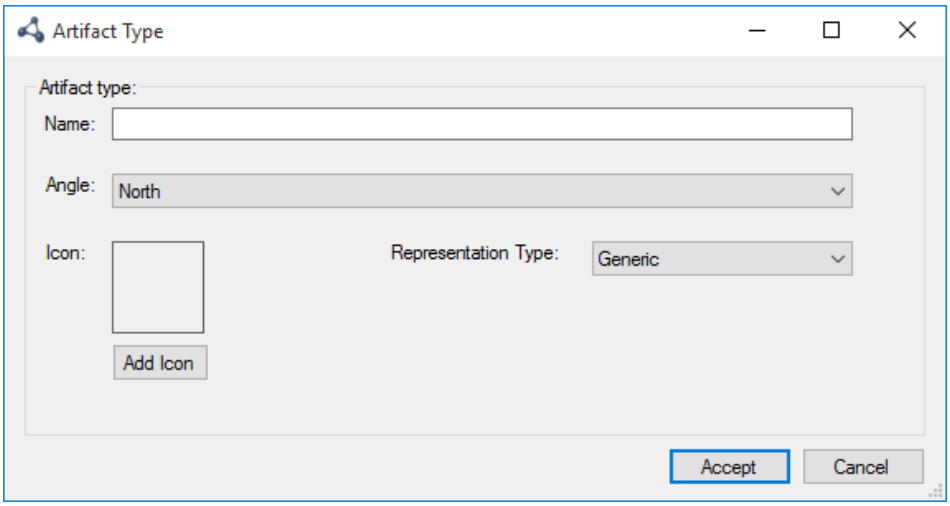

Ilustración 25: Ventana para nuevos tipos de artefacto

Esta ventana incluye los campos necesarios para crear nuevos tipos de artefactos, estos son:

- Nombre identificador del nuevo tipo de artefacto.
- Angulo de su representación (si la tiene), puede ser:
  - Norte, Sur, Este y Oeste.
  - No rotar.
- El icono a añadir, que se mostrarán en el cuadrado de vista previa de esta ventana.
- El tipo de representación, en caso que el nuevo artefacto que se esté creando sea un tipo de diagrama que luego se pueda seleccionar.

Por último cabe destacar que se incluyen un botón de aceptar y de cancelar en el diálogo, para guardar el nuevo tipo de artefacto, o por el contrario cancelar la operación.

- **Modificar tipo de artefacto:**

  En este caso, no se han realizado modificaciones en cuanto a la estética y reparto de los elementos de este formulario, salvo que en la selección de tipo de artefacto, se han añadido también los nuevos tipos de artefactos que puede crear el usuario. Esta es una mejora sustancial teniendo en cuenta que anteriormente el usuario estaba limitado sólo a unos pocos tipos de artefactos, que además se añadían sólo desde el Knowledge Manager. A continuación se muestra una captura con esta lista de selección desplegada en la ventana de modificar artefacto.

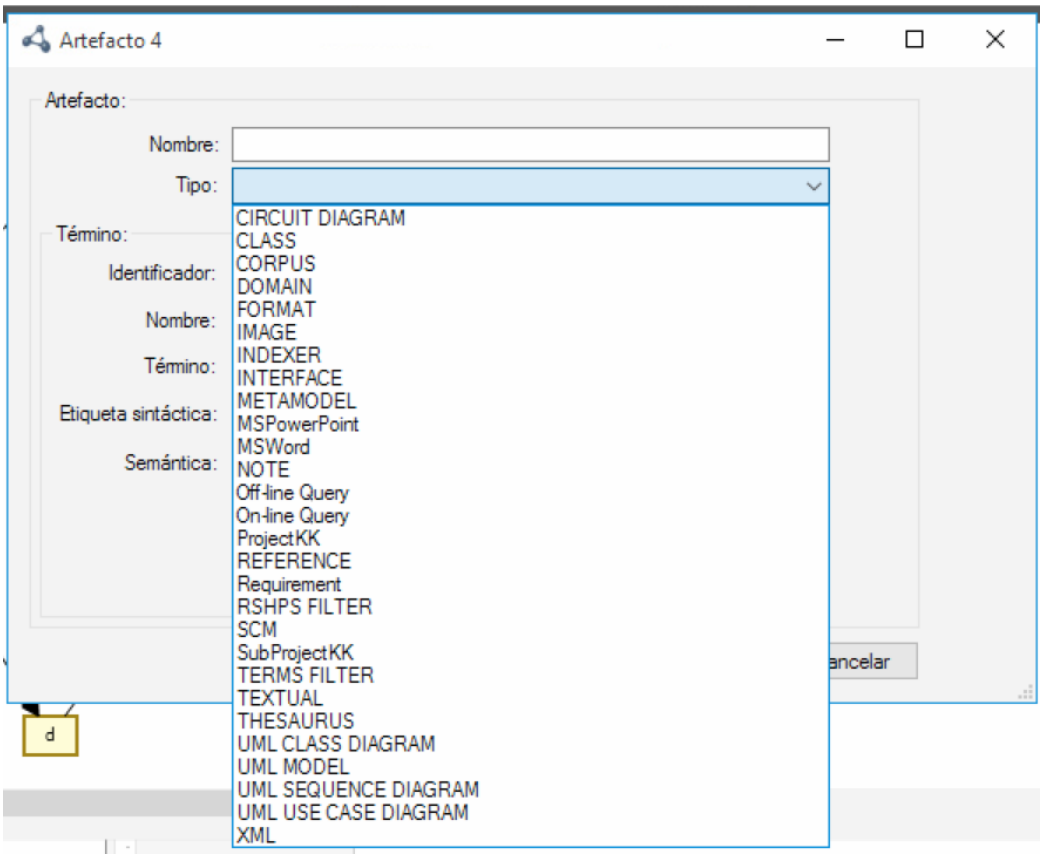

Ilustración 26: Ventana de edición de artefacto con lista desplegada para la selección del tipo de artefacto

- **Seleccionar primer elemento de un ciclo en circuitos eléctricos:**

  En este caso de uso se tratará otra de las nuevas funcionalidades que se incluyen para la edición de elementos de diagramas basados en filosofías de dibujo de circuito eléctrico. Esta consiste en la elección del primer elemento dentro de un circuito eléctrico, editando el elemento en cuestión dentro del canvas. Al marcarlo como primer elemento, las disposición de los elementos dentro del circuito se recolocará en función del que seleccione el usuario al aplicar la acción de layout. Así por ejemplo cuando existen ciclos en un circuito eléctrico (como cuando se cierra por ejemplo) todo los elementos del diagrama se distribuyen para dejar a la izquierda el primer elemento seleccionado por el usuario cuando realice la acción de layout. Esta situación se expresa mejor con una imagen como la que se muestra a continuación:

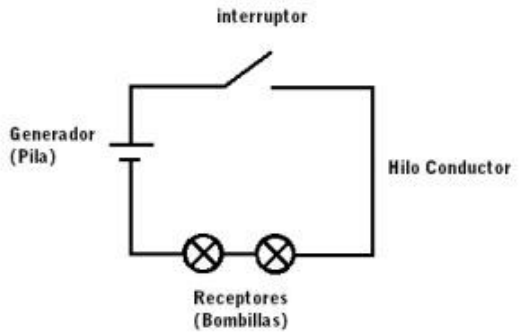

Ilustración 27: Circuito eléctrico cerrado

En el caso mostrado en la ilustración anterior, el primer elemento que seleccionaría el usuario sería el generador, por lo que al hacer layout, este se pondría por defecto a la izquierda definiendo el inicio del circuito cerrado. Esta opción se muestra en la siguiente captura:

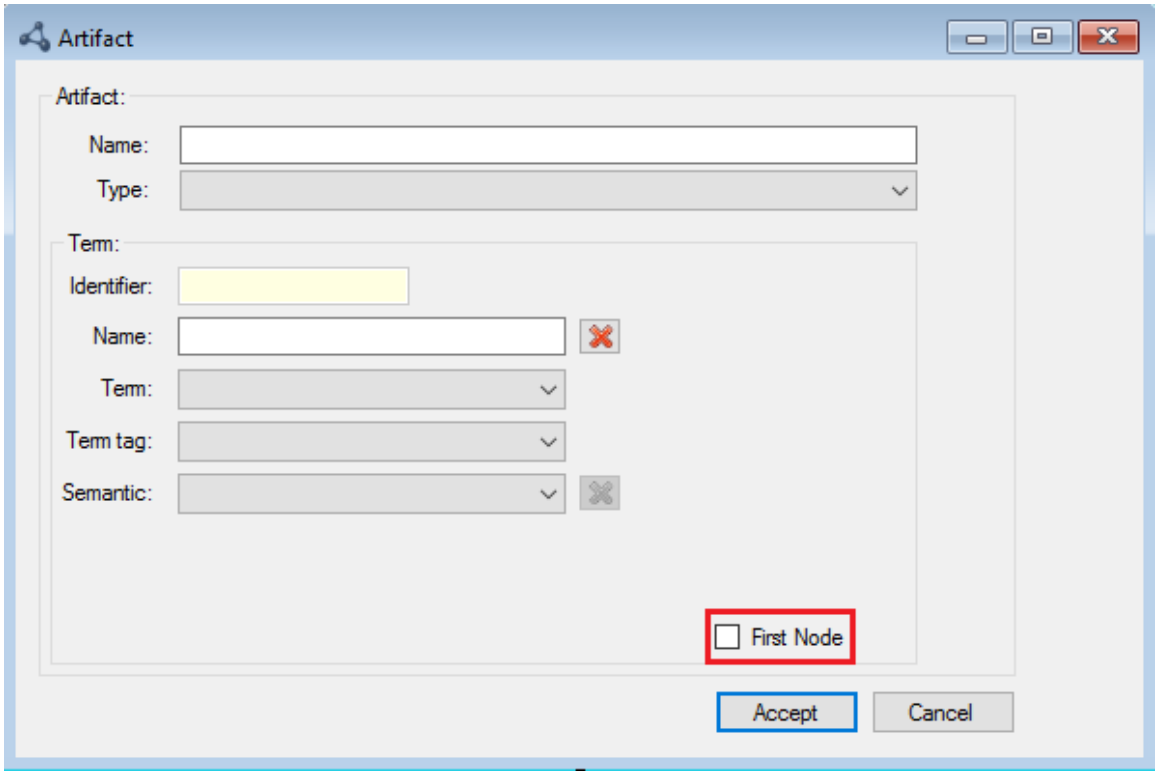

Ilustración 28: Ventana de edición de artefacto dentro de un diagrama de circuito eléctrico

La opción de First Node (primer nodo o primer elemento), se muestra por igual para la ventana de edición de términos, pero tanto como para términos como para artefactos, esta opción sólo se desplegará cuando el tipo de artefacto asociado al diagrama que esté editando el usuario contenga un layout de circuito eléctrico.

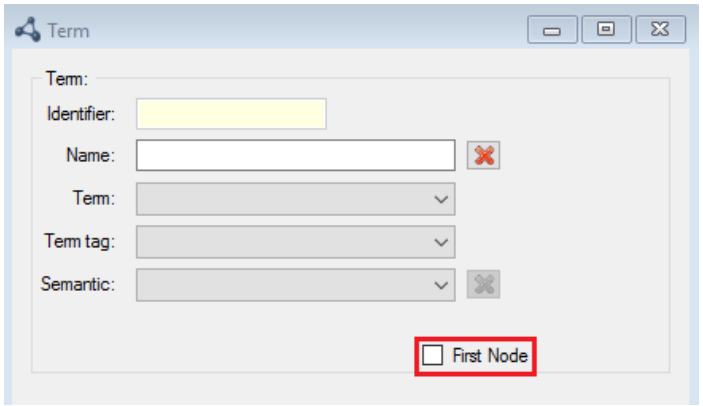

Ilustración 29: Ventana de edición de término en diagramas de circuito eléctrico

4.5 Especificación del plan de pruebas

En este aparatado se define el plan de pruebas, que no es más que una guía para la realización de las pruebas del sistema, y que permite garantizar que el sistema cumple con las necesidades del usuario así como la calidad esperada por el cliente.

4.5.1 Definición del Alcance de las Pruebas

Para el desarrollo de este proyecto de fin de carrera se establecerán pruebas de aceptación, que tiene como objeto la verificación del funcionamiento del sistema con los propios usuarios a fin de obtener su aprobación, dejando de lado otros tipos de pruebas como las de integración o implantación, debido a que será el propio equipo del año pasado quien integre e implante el sistema.

Otro tipo de pruebas que se definirán son las pruebas unitarias, que se elaborarán en la fase de diseño de este documento y que se encargan de verificar el correcto funcionamiento de cada módulo del sistema a un nivel más profundo que las propias pruebas de aceptación.

4.5.2 Definición de Requisitos del entorno de Pruebas

Las pruebas se ejecutarán en el mismo entorno tecnológico que el definido para el desarrollo de este módulo.

4.5.3 Definición de las Pruebas de Aceptación del Sistema

En este punto se especificarán las pruebas a realizar para comprobar que el sistema cumple con todas las necesidades del cliente. De tal manera que el cliente consigue validar su sistema antes de que este pase a su fase de explotación. Por ello este proceso se va desarrollando a lo largo de todo el proyecto para que las que no cumplan con sus expectativas sean modificadas inmediatamente.

A continuación se mostrará la plantilla que se ha definido para recoger las pruebas de aceptación:

PA-##: Nombre que identifica la prueba	
Descripción	Descripción de la prueba.
Escenario	1. Primer paso para la prueba. 2. Segundo… 3. … 4. …
Resultado	Resultado esperado para la prueba.
Requisitos relacionados	RS-YXX (Donde RS es requisito de sistema, Y es el tipo de requisito de sistema y XX el número que identifica al requisito de sistema.

Tabla 66: Plantilla para las tablas de pruebas de aceptación de sistema

A continuación se muestran todas las pruebas para probar el correcto funcionamiento del sistema:

PA-01: Comprobar edición de layout asociado a un diagrama	
Descripción	Se comprueba que se puede cambiar la filosofía de dibujo asociada a un diagrama en el modo nativo.
Escenario	1. Comprobar que el diagrama tiene asignado un layout por defecto, sino realizar los pasos del 2 al 5 y luego volver al 2 con seleccionando otro layout. 2. Seleccionar el cambio de tipo de gráfico. 3. Dentro de la ventana seleccionar el tipo de artefacto que tenga asociada una filosofía de dibujo (por ejemplo se ofrece al usuario “CIRCUIT DIAGRAM”). 4. Pulsar aceptar. 5. Aplicar layout.
Resultado	La etiqueta que indica el tipo de diagrama, debe haber cambiado al nombre de layout que el usuario le ha asignado, además al aplicar el layout se muestra que el diagrama cambia y por tanto se ha asignado otro tipo de diagrama.
Requisitos relacionados	RS-F007, RS-F009, RS-010

Tabla 67: PA-01: Comprobar edición de layout asociado a un diagrama

PA-02: Comprobar acción de layout para circuito eléctrico en modo nativo	
Descripción	Se comprueba que el diagrama en modo nativo se puede recolocar como un diagrama de circuito eléctrico
Escenario	1. Seleccionar el cambio de tipo de gráfico. 2. Dentro de la ventana seleccionar el tipo de artefacto que tenga asociada una filosofía de circuito eléctrico (por defecto se ofrece al usuario “CIRCUIT DIAGRAM”). 3. Pulsar aceptar. 4. Clicar en el botón de layout.
Resultado	Cuando se pulsa el botón de layout en el modo nativo, el diagrama que se mostraba originalmente ha cambiado la posición de sus elementos siguiendo la disposición definida para la filosofía de dibujo de circuito eléctrico.
Requisitos relacionados	RS-F009, RS-F010, RS-F014

Tabla 68: PA-02: Comprobar acción de layout para circuito eléctrico en modo nativo

PA-03: Comprobar acción de layout para circuito eléctrico en modo transformado	
Descripción	Se comprueba que el diagrama se puede recolocar como un diagrama de circuito eléctrico
Escenario	1. Seleccionar el cambio de tipo de gráfico. 2. Dentro de la ventana seleccionar el tipo de artefacto que tenga asociada una filosofía de circuito eléctrico (por defecto se ofrece al usuario “CIRCUIT DIAGRAM”). 3. Pulsar aceptar. 4. Clicar en el botón de transformación. 5. En la ventana de transformación, pulsar el botón de layout.
Resultado	Cuando se pulsa el botón de layout en el modo transformado, el diagrama que se mostraba originalmente ha cambiado la posición de sus elementos siguiendo la disposición definida para la filosofía de dibujo de circuito eléctrico, dando igual que por ejemplo se muestren las representaciones de los iconos al realizar la acción de layout.
Requisitos relacionados	RS-F009, RS-F010, RS-F014

Tabla 69: PA-03: Comprobar acción de layout para circuito eléctrico en modo transformado

PA-04: Comprobar edición de layout asociado a un diagrama sólo en modo nativo	
Descripción	Se comprueba que el layout asociado al diagrama sólo se puede cambiar en modo nativo
Escenario	1. Seleccionar el botón de transformar.
Resultado	El botón de cambio de tipo de diagrama no se muestra en la ventana del modo transformado.
Requisitos relacionados	RS-F012

Tabla 70: PA-04: Comprobar edición de layout asociado a un diagrama sólo en modo nativo

PA-05: Comprobar que se conservan los cambios desde el modo nativo al transformado	
Descripción	Se comprueba que las modificaciones realizadas por el usuario sobre el diagrama se conservan en el modo transformado.
Escenario	1. Arrastra varios elementos al canvas 2. Cambiarlos de posición. 3. Pulsar el botón de transformación.
Resultado	Al transformar el diagrama desde su modo nativo al modo transformado, se preservan todas las posiciones de todos los nodos y de todas las relaciones (por ejemplo los puntos que definen las relaciones, manualmente por acción de un usuario, pueden formar cierto ángulo en lugar de una línea recta). La única modificación que se mostrará en los nodos serán las representaciones de los mismos si estos tienen asociados un tipo de artefacto con representación.
Requisitos relacionados	RS-F011

Tabla 71: PA-05: Comprobar que se conservan los cambios desde el modo nativo al transformado

PA-06: Comprobar que se conservan los cambios desde el modo transformado al nativo	
Descripción	Se comprueba que las modificaciones realizadas por el usuario sobre el diagrama se conservan en el modo transformado.
Escenario	1. Arrastra varios elementos al canvas 2. Pulsar el botón de transformación. 3. En la nueva ventana de transformación, cambiarlos de posición. 4. Pulsar aceptar.
Resultado	Cuando se pulsa aceptar en la ventana de transformación, todos los nuevos cambios que el usuario ha realizado sobre la posición de nodos y relaciones del diagrama se preservan en el modo nativo. Así el diagrama nativo anterior tendrá todos los cambios que el usuario ha aplicado en el modo transformado.
Requisitos relacionados	RS-F011

Tabla 72: PA-06: Comprobar que se conservan los cambios desde el modo transformado al nativo

PA-07: Comprobar que se añade correctamente un nuevo tipo de artefacto	
Descripción	Se comprueba que el usuario puede añadir correctamente un nuevo tipo de artefacto.
Escenario	1. Arrastra un artefacto al canvas 2. Pulsar el botón de añadir nuevo tipo de artefacto. 3. Rellenar el formulario con los datos como el nombre del nuevo tipo de artefacto, la orientación de su representación (si la tiene), la imagen que funciona como su representación, así como la filosofía de dibujo asociada al tipo de artefacto (de nuevo, si la tiene). 4. Pulsar aceptar. 5. Editar el artefacto que se ha añadido al canvas. 6. Desplegar los tipos de artefactos que puede tener.
Resultado	Cuando se añade el nuevo tipo de artefacto, este se serializa en la BBDD y al editar el artefacto que se ha añadido al canvas, y seleccionar su tipo de artefacto, el nuevo tipo de artefacto que se ha añadido aparecerá mostrando que se ha añadido correctamente.
Requisitos relacionados	RS-F001, RS-F004, FS-006

Tabla 73: PA-07: Comprobar que se añade correctamente un nuevo tipo de artefacto

PA-08: Editar artefacto para asignarle un tipo de artefacto con representación y comprobar que se muestra la misma	
Descripción	Se comprueba que el usuario puede añadir correctamente un nuevo tipo de artefacto, asignárselo a un artefacto, y que al transformar se muestre el icono del tipo de artefacto correctamente.
Escenario	1. Arrastra un artefacto al canvas 2. Pulsar el botón de añadir nuevo tipo de artefacto. 3. Rellenar el formulario con los datos como el nombre del nuevo tipo de artefacto, la orientación de su representación (si la tiene), una imagen a forma de icono, así como la filosofía de dibujo asociada al tipo de artefacto (de nuevo, si la tiene). 4. Pulsar aceptar. 5. Editar el artefacto que se ha añadido al canvas. 6. Desplegar los tipos de artefactos que puede tener. 7. Seleccionar el tipo de artefacto que se acaba de añadir. 8. Pulsar aceptar en la ventana de edición del artefacto. 9. Comprobar que el diagrama tiene una filosofía de dibujo asociada que permita la representación de iconos para los nodos del diagrama (Por ejemplo asignando un layout de circuito eléctrico). 10. Transformar el diagrama.
Resultado	Cuando se añade el nuevo tipo de artefacto con representación, y este se asigna a un artefacto del diagrama, si se transforma en una filosofía de dibujo que permita representaciones (como la de circuito eléctrico) se mostrará la representación que se ha añadido previamente.
Requisitos relacionados	RS-F001, RS-F002, RS-F003

Tabla 74: PA-08: Editar artefacto para asignarle un tipo de artefacto con representación y comprobar que se muestra la misma

PA-09: Comprobar que las imágenes de los artefactos rotan en el modo transformado tras aplicar layout	
Descripción	Se comprueba que las representaciones de los artefactos añadidos a un diagrama rotan en el modo de transformación cuando se encuentran en un diagrama con una filosofía de dibujo basada en conexión (Por ejemplo circuito eléctrico).
Escenario	1. Arrastra varios artefacto al canvas 2. Establecer relaciones entre ellos. 3. Pulsar el botón de añadir nuevo tipo de artefacto. 4. Rellenar el formulario con los datos como el nombre del nuevo tipo de artefacto, la orientación de su representación, una imagen a forma de icono, así como la filosofía de dibujo asociada al tipo de artefacto (de nuevo, si la tiene). 5. Pulsar aceptar. 6. Añadir unos artefactos más 7. Editar los artefactos que se han añadido al canvas. 8. Desplegar los tipos de artefactos que puede tener. 9. Seleccionar uno de los tipos de artefacto que se acaban de añadir. 10. Pulsar aceptar en la ventana de edición del artefacto. 11. Comprobar que el diagrama tiene una filosofía de dibujo asociada que permita la representación de iconos para los nodos del diagrama (Por ejemplo asignando un layout de circuito eléctrico). 12. Transformar el diagrama. 13. En el modo transformado (o antes de transformar el diagrama) aplicar la acción de layout.
Resultado	Cuando se añade el nuevo tipo de artefacto con representación, y este se asigna a un artefacto del diagrama, si se transforma en una filosofía de dibujo que permita representaciones (como la de circuito eléctrico) se mostrará la representación que se ha añadido previamente. Además si este artefacto tiene relaciones entrantes y salientes, y los puntos de la relación que conectan por los extremos al artefacto forman un ángulo, la imagen rotará según el sentido que tiene asignado para estar en línea con estos puntos.
Requisitos relacionados	RS-F001, RS-F002, RS-F003, RS-F005

Tabla 75: PA-09: Comprobar que las imágenes de los artefactos rotan en el modo transformado tras aplicar layout

PA-10: Comprobar que las imágenes de los artefactos rotan en el modo transformado cuando el usuario mueve los nodos	
Descripción	Se comprueba que las representaciones de los artefactos añadidos a un diagrama rotan en el modo de transformación cuando se encuentran en un diagrama con una filosofía de dibujo basada en conexión (Por ejemplo circuito eléctrico), y se define un eje entre los puntos de las relaciones que se conectan con los artefactos tras la recolocación de los mismos por el usuario en el diagrama
Escenario	1. Arrastra varios artefacto al canvas 2. Establecer relaciones entre ellos. 3. Pulsar el botón de añadir nuevo tipo de artefacto. 4. Rellenar el formulario con los datos como el nombre del nuevo tipo de artefacto, la orientación de su representación, una imagen a forma de icono, así como la filosofía de dibujo asociada al tipo de artefacto (de nuevo, si la tiene). 5. Pulsar aceptar. 6. Añadir unos artefactos más 7. Editar los artefactos que se han añadido al canvas. 8. Desplegar los tipos de artefactos que puede tener. 9. Seleccionar uno de los tipos de artefacto que se acaban de añadir. 10. Pulsar aceptar en la ventana de edición del artefacto. 11. Comprobar que el diagrama tiene una filosofía de dibujo asociada que permita la representación de iconos para los nodos del diagrama (Por ejemplo asignando un layout de circuito eléctrico). 12. En el modo transformado (o antes de transformar el diagrama) desplazar algunos artefactos a nuevas posiciones. 13. Transformar el diagrama.
Resultado	Cuando se añade el nuevo tipo de artefacto con representación, y este se asigna a un artefacto del diagrama, si se transforma en una filosofía de dibujo que permita representaciones (como la de circuito eléctrico) se mostrará la representación que se ha añadido previamente. Además si este artefacto tiene relaciones entrantes y salientes, y los puntos de la relación que conectan por los extremos al artefacto forman un ángulo, la imagen rotará según el sentido que tiene asignado para estar en línea con estos puntos, aunque esté ángulo no sea producto de la acción de aplicar un layout, sino más bien la edición de la posición de un nodo por parte del usuario.
Requisitos relacionados	RS-F001, RS-F002, RS-F003, RS-F011, RS-F013

Tabla 76: PA-10: Comprobar que las imágenes de los artefactos rotan en el modo transformado cuando el usuario mueve los nodos

PA-11: Comprobar campos obligatorios de un nuevo tipo de artefacto	
Descripción	Se comprueba que los campos obligatorios a la hora de rellenar la información de un nuevo tipo de artefacto son correctos
Escenario	1. Pulsar el botón de nuevo tipo de artefacto. 2. En la ventana que se despliega, introducir el nombre de un artefacto existente en la BBDD. 3. Pulsar aceptar. 4. Seleccionar una orientación para la representación de una imagen y no incluir dicha representación. 5. Pulsar aceptar.
Resultado	Cuando se añade el nuevo tipo de artefacto, las únicas condiciones que se requieren para que se añada correctamente son que no exista otro artefacto con su mismo nombre, y que la orientación para su representación vaya de la mano con esta representación. En caso contrario se mostrará un mensaje de error al usuario.
Requisitos relacionados	RS-F08

Tabla 77: PA-11: Comprobar campos obligatorios de un nuevo tipo de artefacto

4.5.4 Matriz de trazabilidad entre Pruebas de aceptación y Requisitos.

A continuación se muestra una matriz de trazabilidad entre las pruebas de aceptación y los requisitos para mostrar que las pruebas cubren con las necesidades del cliente.

IDs	PA-01	PA-02	PA-03	PA-04	PA-05	PA-06	PA-07	PA-08	PA-09	PA-10	PA-11
RS-F001							√	√	√	√	
RS-F002								√	√	√	
RS-F003								√	√	√	
RS-F004							√				
RS-F005									√		
RS-F006							√				
RS-F007	√										
RS-F008											√
RS-F009	√	√	√								
RS-F010	√	√	√								
RS-F011					√	√				√	
RS-F012				√							
RS-F013										√	
RS-F014		√	√								

Tabla 78: Matriz de trazabilidad entre Pruebas de aceptación y Requisitos

5. Diseño

En este apartado de diseño se pretende resolver el problema descrito y modelado en el apartado de Análisis de este documento.

Este apartado por lo tanto asienta las bases en la que se sustenta la implementación de este proyecto, lo que facilita en gran medida el desarrollo del mismo. Así se proporciona la estructura básica de este sistema, los diferentes componentes que lo conforman y las relaciones entre ellos.

A continuación se describirá la arquitectura del sistema para los objetivos especificados para este proyecto.

5.1 Definición de la arquitectura del sistema

En este apartado se definirán todos los aspectos relacionados con la arquitectura de este sistema.
Cabe destacar que la arquitectura definida para esta mejora es prácticamente idéntica a la definida el año pasado, por lo que no se incidirá en profundidad en la misma. Para cualquier consulta se incluirá en la bibliografía las referencias del material documental de la etapa anterior de desarrollo de este módulo.

5.1.1 Definición de los niveles de arquitectura

Entendiendo este proyecto como una mejora del módulo que desarrollaron Pablo Sánchez Pérez, Víctor Sacristán Tejudo, y Alejandro Fragueiro Oliva, y teniendo en cuenta que todo lo desarrollado en la etapa anterior debe respetarse y no modificarse salvo previo permiso al cliente, se ha decidido mantener el mismo patrón de arquitectura basado en una arquitectura Modelo-Vista-Controlador (MVC).

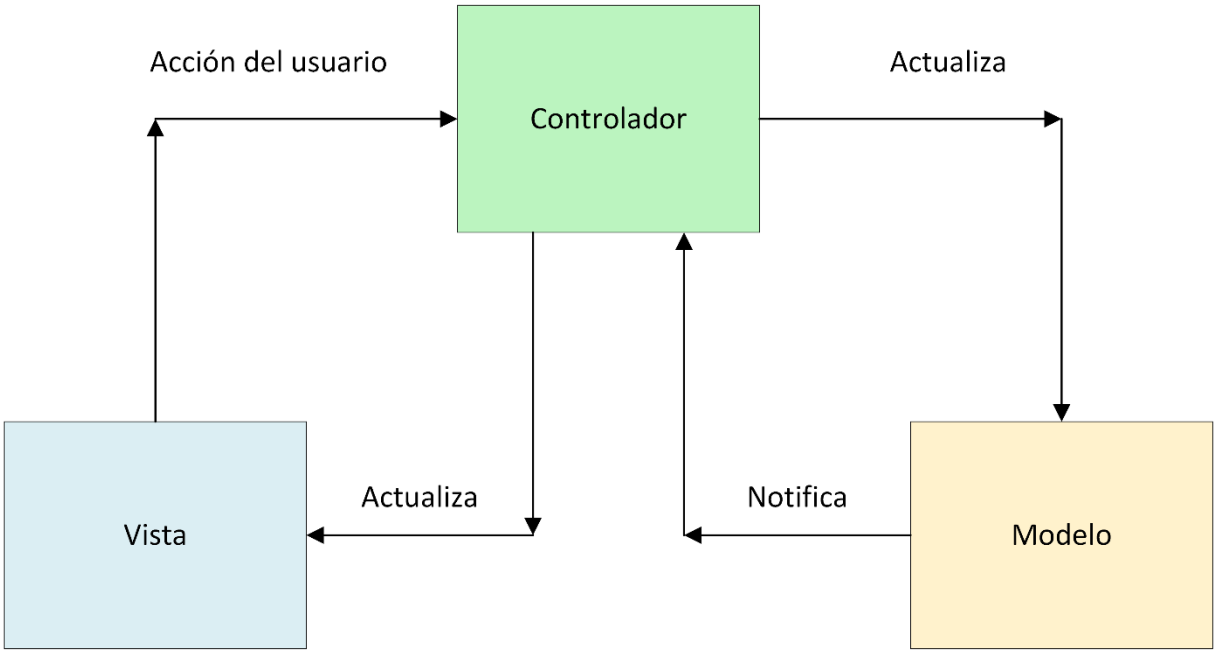

Ilustración 30: Arquitectura Modelo Vista Controlador

Esta arquitectura, MVC, es un patrón de arquitectura software extensamente usado, que se basa en la separación de los datos y la lógica de negocio de una aplicación que además ofrece al usuario una interfaz para interactuar. A continuación se detallarán estas capas:

- **Vista:** Es la capa que presenta la información que se ofrece al usuario, siendo además la interfaz de comunicación del usuario con el sistema (al menos en este caso). Está separada del modelo y el controlador, y se encarga de recoger las entradas por parte del usuario y de presentar la información que le comunica el modelo como salida.
- **Controlador:** Corresponde a la capa intermedia entre la vista y el modelo. Su fin es recoger toda la funcionalidad del sistema, siendo parte de esta funcionalidad responder a eventos (normalmente acciones del usuario recogidas por la vista), e invocar peticiones al modelo cuando se realiza una solicitud de información (cualquier actualización en la BBDD por ejemplo). También se encarga de enviar modificaciones a la vista, por ejemplo, después de alguna actualización en el modelo.
- **Modelo:** En esta capa reside toda la información del sistema, incluyendo también funciones para poder gestionar nuevas inserciones de información, modificaciones y borrados. Otra función muy importante para esta capa es la gestión de privilegios de acceso y modificación de la información del sistema que almacena, siendo estos definidos por el usuario normalmente en la parte de análisis del sistema. Todas las peticiones para la gestión de información llegan desde el controlador.

  A continuación se mostrará una imagen con la arquitectura que se ha descrito anteriormente aplicada a este módulo que se está mejorando:

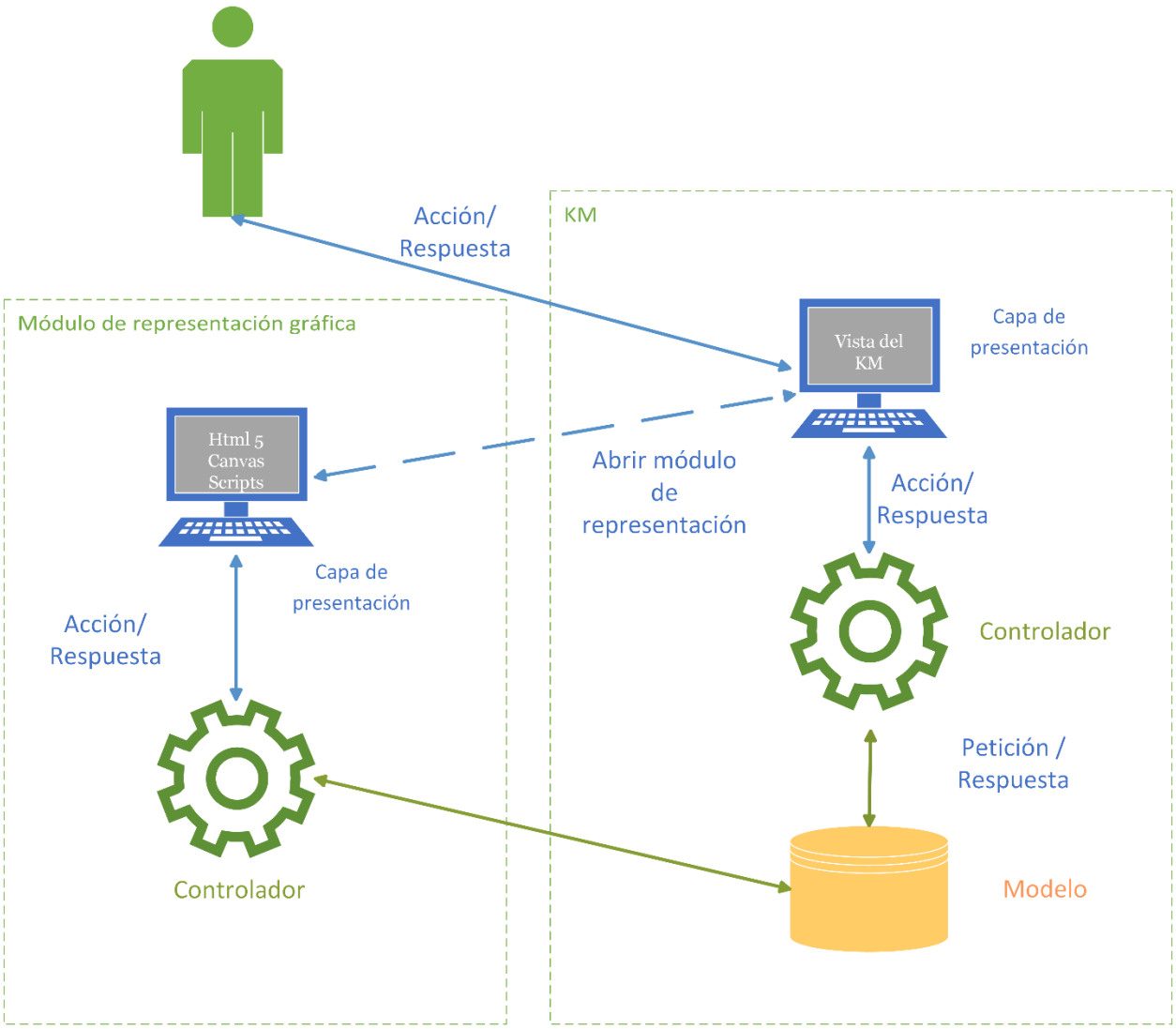

Ilustración 31: Niveles de arquitectura aplicados a este sistema

Tal como se muestra en esta imagen, este módulo presenta información directamente del modelo de datos que se usa para almacenar la información del Knowledge Manager, y el sistema se despliega cuando el usuario decide abrirlo también desde el Knowledge Manager. Cabe destacar que este diseño intenta abstraer lo mejor posible la arquitectura de este sistema, y es que el controlador del módulo es una capa muy compleja, ya que es fundamental que sea capaz de procesar información procedente de una interfaz gráfica a un modelo de datos concreto que es usado a su vez por otra aplicación.

5.1.2 Especificación de estándares y normas de diseño y construcción

En este punto se especifican los diferentes estándares, normas y recomendaciones que se usarán en el diseño y desarrollo de este proyecto de fin de carrera.

Debido a que este proyecto es una mejora, sabiendo que las personas que se encargarán del mantenimiento de este sistema son las mismas que comenzaron el desarrollo de este módulo y también por una petición expresa del cliente, se mantendrán los mismos estándares establecidos en la etapa anterior de desarrollo. A continuación citaré los mismos de la memoria de Víctor Sacristán Tejudo:

- ***Atributos**: el nombre de los atributos se escriben con la primera palabra en minúscula. En el caso de estar compuesta por más de una palabra, las demás se escribirán con la primera letra en mayúscula.*
- ***Constantes**: se escriben en mayúscula. Si el nombre de la constante abarca más de una palabra, se separan mediante un guion bajo.*
- ***Funciones**: el nombre de las funciones es similar al de los atributos. Se escribe la primera palabra en minúscula, exceptuando las de la inicial de las demás palabras por las que pueda estar compuesta.*
- ***Clases**: se escriben en minúscula, exceptuando la primera letra de cada palabra, la cual se escribe en mayúscula.*

Además de dichas configuraciones, se tendrán en cuenta algunas nuevas que no se especificaron:

- **Comentarios:** Se incluirán comentarios en todos los métodos (funciones) desarrollados durante esta mejora. Todos estos comentarios irán encima de la declaración del método. Además para explicar la lógica que se ha implementado, será necesario añadir dentro del propio método todos los comentarios que se estimen oportunos, facilitando así la comprensión de los mismos para los futuros desarrolladores que seguirán mejorando este sistema.
- **Funciones:** Además de la nomenclatura, ninguna función debe tener una extensión mayor de 15 líneas de código, ya que así es mucho más sencillo trazar cualquier fallo de la aplicación. Si alguna tiene una longitud superior a 15 líneas, se deberá separar en dos métodos para así mantener un código comprensible y ordenado.

- **Configuraciones de sangrías:** Se usarán las configuraciones proporcionadas por The Reuse Company para los diferentes sangrados y organizaciones de las líneas de código en C#. De igual manera, y aunque no sean aplicables desde el entorno de programación seleccionado (Visual Studio 2010), el código JavaScript que se modifique o desarrolle debe seguir estás configuraciones, las cuales se aplicarán manualmente.

5.1.3 Identificación de los subsistemas de diseño

En este apartado se van a identificar las particiones en las que se divide el sistema para poder simplificar enormemente el desarrollo de las mejoras sobre este módulo. Estas divisiones ya se especificaron en el apartado de Identificación de subsistemas de análisis, y se van a usar los mismos para este apartado de diseño.

5.2 Diseño de clases

En este punto se describirán las clases de diseño que intervienen en este sistema, como se relacionan, así como las funciones y atributos con los que cuentan.

Como se indicó anteriormente en la parte de análisis del sistema de este documento, el diseño de clases no es un diseño al uso al no ser un proyecto creado de cero, sino que se recogerán los cambios aplicados sobre las clases existentes, así como alguna nueva clase que ha sido necesaria crear para llegar a cubrir la funcionalidad requerida por el cliente.
Ha sido una tarea realmente ardua, averiguar cómo se debería documentar el análisis de las clases implicadas en el sistema, así como documentar el diseño de las mismas, puesto que no se posee experiencia acerca de la gestión documental en proyectos que se van a mejorar. Así podría ser interesante incluir en la documentación referente al estándar de Métrica V3, un anexo de cómo abordar la documentación de proyectos que se mejoran.

A continuación se mostrarán dos imágenes, la primera es el diseño de clases al completo creado en la etapa de desarrollo anterior de este sistema, incluyendo los 3 proyectos realizados por los anteriores desarrolladores.

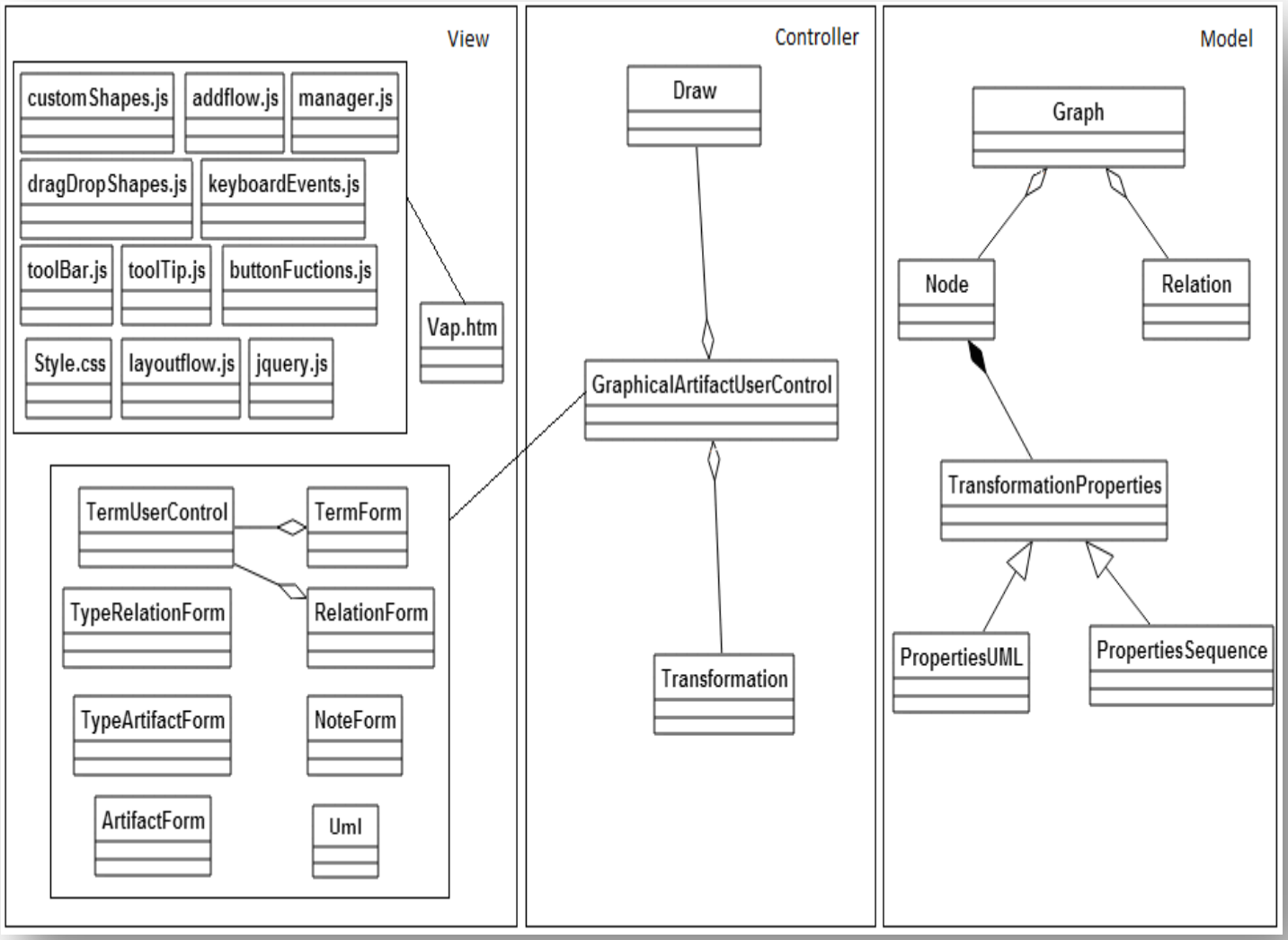

Ilustración 32: Diagrama de clases original

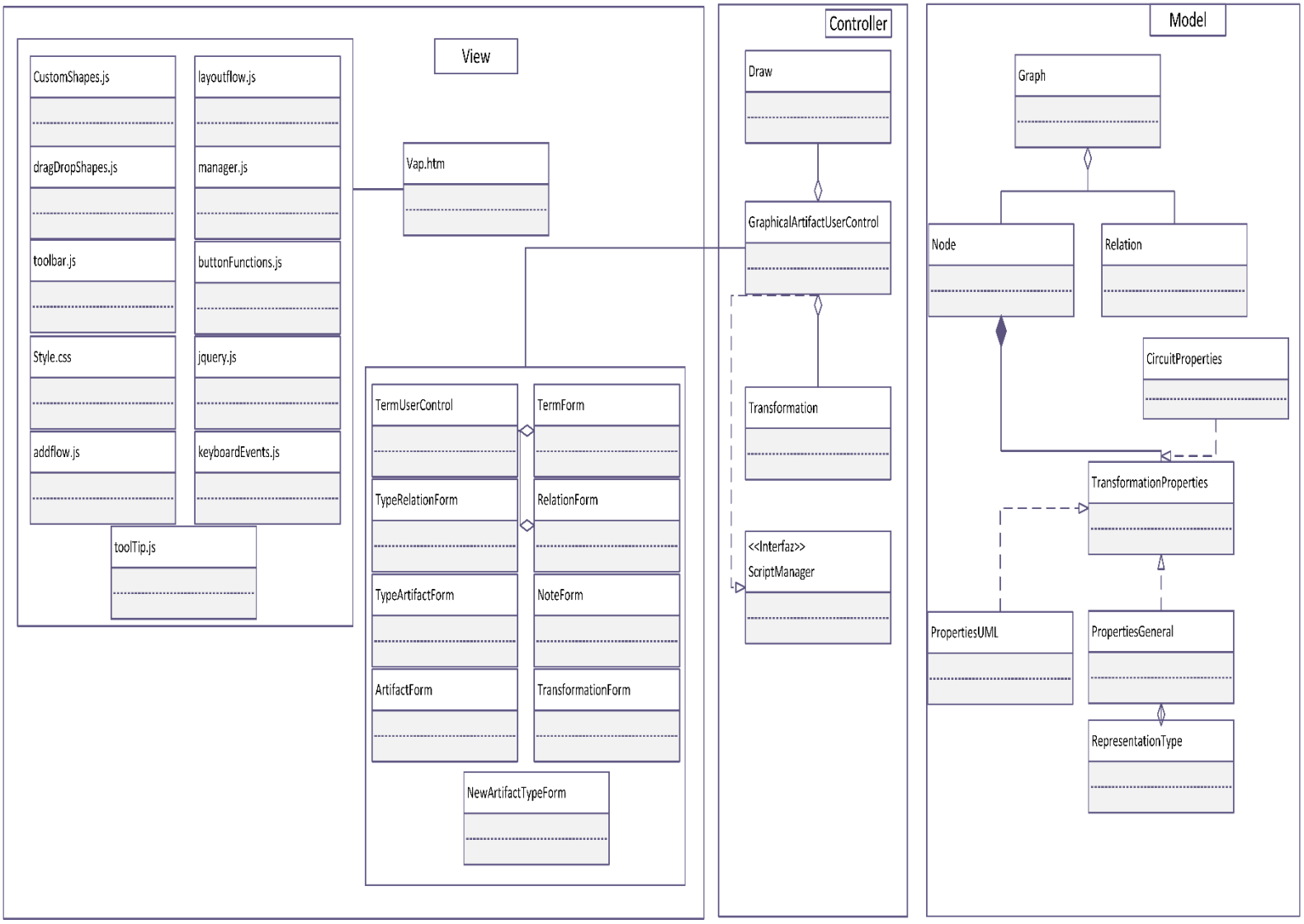

Ilustración 33: Nuevo diagrama de clases

Como se puede apreciar comparando ambos diagramas, estos son prácticamente iguales, siendo las mayores diferencias los cambios añadidos para la nueva clase que permite almacenar nuevos tipos de artefactos (NewArtifactTypeForm) y la interfaz de ScriptManager, que aunque existía en el proyecto anterior no se recogió en el diagrama, siendo está una interfaz tremendamente importante al permitir comunicar eventos del controlador basado en C# con las clases JavaScript para acciones de la capa de la vista.

También son destacables las nuevas clases que se han creado para el modelo de datos. Son CircuitProperties y RepresentationType, ya que PropertiesGeneral es la modificación de la clase que definía el modelo para diagramas de secuencia y que ahora albergará cualquier tipo de representación que no requiera datos adicionales como ocurre con PropertiesUML y

CircuitProperties (respectivamente sirviendo la primera de modelo de datos para UML y la segunda para circuitos eléctricos). Cabe destacar la presencia de RepresentationType, que será una clase fundamental en esta mejora ya que recoge todo el almacenamiento de imágenes, permitiendo serializarlas en el modelo de datos.

5.2.1 Identificación de clases

En este punto se identificarán las clases nuevas y las mejoras aplicadas sobre las existentes según las necesidades recogidas en el punto de Análisis de clases. A continuación se enumerarán estas clases agrupadas según la capa de la arquitectura del sistema a la que pertenezcan, mostrándose además cuáles serán clases nuevas y cuáles son clases que se han modificado:

- **Clases de entidad:**
  - **Nuevas:**
    - CircuitProperties.
    - PropertiesGeneral.
    - RepresentationType.
  - **Mejoras sobre clases existentes:**
    - Node.
    - Relation.
    - Graph.
- **Clases de interfaz:**
  - **Nuevas:**
    - NewArtifactTypeForm.
  - **Mejoras sobre clases existentes:**
    - ArtifactForm.
    - TermForm.
    - TypeArtifactForm.
- **Clases de control:**
  - **Mejoras sobre clases existentes:**
    - GraphicalUserControl.
    - Transformation.

5.2.2 Especificación de clases

En este apartado se detallarán todas las clases que se han diseñado para la consecución de este proyecto, describiendo las propiedades que tiene cada una. De nuevo se mostrarán las que son nuevas y las que son simples mejoras sobre clases existentes. Para ello se especificará una plantilla igual a la del punto del Análisis de clases, sólo que cambiando el identificador de la misma por CLD (Clase de Diseño).

5.2.2.1 Clases de entidad

- **Clases nuevas:**

CLD-01: CircuitProperties	
Responsabilidades	Se encarga de interactuar con la capa de datos (modelo), para recoger los datos relacionados con las propiedades de un elemento de un circuito eléctrico, permitiendo aplicar la implementación de los algoritmos presentes en el apartado de Proceso de selección de algoritmos para dibujar diagramas de circuitos eléctricos.
Atributos	❖ `private List<Relation> inEdges` Lista de objetos relación que entran al nodo. ❖ `private List<Relation> outEdges` Lista de objetos relación que salen del nodo. ❖ `private int lowlink` Propiedad para saber si se han visitado cuando intentamos colocar los links en un diagrama eléctrico. ❖ `private int index` Variable con el orden del nodo en el layout eléctrico.
Funcionalidades	-

Tabla 79: CLD-01: CircuitProperties

CLD-02: PropertiesGeneral	
Responsabilidades	Se encarga de interactuar con la capa de datos (modelo), para recoger los datos relacionados con las representaciones que se cargan en los nodos del diagrama, mostrando estas imágenes en la transformación del diagrama.
Atributos	❖ `private string image` Variable que contiene la imagen codificada en un string de Base 64. ❖ `private int width` Variable que contiene el ancho de la imagen en puntos. ❖ `private int height` Variable que contiene la altura de la imagen en puntos. ❖ `private bool direction` Variable que se usa cuando el diagrama es de circuito eléctrico, indicando dónde debe colocarse la información acerca del nodo (A la izquierda si es el primer nodo que desciende del circuito, a las derecha si es el último, o arriba en el resto de los casos).
Funcionalidades	-

Tabla 80: CLD-02: PropertiesGeneral

CLD-03: RepresentationType	
Responsabilidades	Se encarga de interactuar con la capa de datos (modelo), para recoger los datos relacionados con las representaciones que se cargan en los tipos de artefactos nuevos, asociados a nodos del diagrama, que crea el usuario a través de los formularios nuevos.
Atributos	❖ `private string artifactTypeName` Variable que contiene el nombre del nuevo tipo de artefacto. ❖ `private Bitmap` Variable que contiene la imagen desde su formato original. ❖ `private string image` Variable que contiene la imagen después de codificarla desde un array de bytes a un string en Base 64. ❖ `private byte[] bytesImg` Array de bytes auxiliar que contiene la imagen para convertirla desde Bitmap a string en Base 64 (es necesario usar este array para la conversión). ❖ `private double angle` Variable que contiene el ángulo del sentido de la imagen. ❖ `private int layout` Variable que indica el tipo de filosofía de dibujo asociado al diagrama donde se va a representar la imagen.
Funcionalidades	❖ `public byte[] bitmap2Byte(Bitmap img)` Función que convierte una imagen almacenada en un Bitmap en un array de bytes ❖ `public Image byteArrayToImage(byte[] byteArrayIn)` Función que convierte un array de bytes en un objeto tipo Image, es decir devuelve la imagen para poder mostrarse (por ejemplo en el selector de imagen de nuevo tipo de artefacto). ❖ `public string imageToBase64(byte[] array)` Función que convierte un array de bytes en un string en Base 64. ❖ `public Bitmap rotateImage(Bitmap b, float angle)` Función que rota la imagen de un bitmap los grados que se estipulen en el float que recibe como parámetro.

Tabla 81: CLD-03: RepresentationType

- **Clases sobre las que se incorporan mejoras:**

CLD-04: Graph	
Responsabilidades	Se encarga de interactuar con la capa de datos (modelo), para recoger los datos relacionados con el gráfico que se representa.
Atributos	❖ `private int representationType` Atributo que indica el tipo de filosofía de dibujo que tiene asociada el diagrama. ❖ `private int down` Atributo que indica el caso para la colocación de los elementos del circuito eléctrico como se detalla en el punto de Configuraciones de circuitos eléctricos según las necesidades del cliente. ❖ `private int postOrderNumber` Atributo que indica el orden general de los nodos al aplicar el layout de circuito eléctrico según el algoritmo de Tarjan. ❖ `private Stack<Node> s` Pila de nodos usada para el algoritmo de Tarjan. ❖ `private List<List<Node>> stronglyConnectedComponents` Lista de componentes fuertemente conexas. El algoritmo de Tarjan viene especificado en el punto de Selección de algoritmos para detectar ciclos.
Funcionalidades	Tiene funciones que no se han mejorado y se detallan en la memoria de Víctor Sacristán Tejudo.

Tabla 82: CLD-04: Graph

CLD-05: Relation	
Responsabilidades	Se encarga de interactuar con la capa de datos (modelo), para recoger los datos relacionados con el gráfico que se representa.
Atributos	❖ `private string leftPoints` Atributo que almacena en un string las coordenadas x de todos los puntos de una relación, separados por ";" para poder aplicar la función Trim (), que separa las cadenas de caracteres que configuran un string detectando un determinado patrón. ❖ `private string topPoints` Atributo que almacena en un string las coordenadas y de todos los puntos de una relación, separados por ";" para poder aplicar la función Trim (), que separa las cadenas de caracteres que configuran un string detectando un determinado patrón.

	❖ `private bool isInCycle` Atributo que indica si una relación se encuentra en un ciclo o no. ❖ `private bool turned` Atributo que indica si una relación ha sido tratada al aplicar el algoritmo de Tarjan. ❖ `private bool firstTarjan` Atributo que indica si la relación es la primera en orden en un ciclo determinado detectado por el algoritmo de Tarjan. ❖ `private bool lastTarjan` Atributo que indica si la relación es la última en orden en un ciclo determinado detectado por el algoritmo de Tarjan.
Funcionalidades	Tiene funciones que no se han mejorado y se detallan en la memoria de Víctor Sacristán Tejudo.

Tabla 83: CLD-05: Relation

CLD-06: Node	
Responsabilidades	Se encarga de interactuar con la capa de datos (modelo), para recoger los datos relacionados con los nodos del diagrama.
Atributos	❖ `private bool isFirst` Variable booleana que permite saber si el nodo es el primero dentro de un diagrama de un circuito eléctrico. ❖ `private bool isLast` Variable booleana que permite saber si el nodo es el último dentro de un diagrama de un circuito eléctrico. ❖ `private double left` Variable que indica la coordenada x que ocupa el nodo en el diagrama, sirve para actualizar la posición entre el modo nativo y transformado y al contrario. ❖ `private double top` Variable que indica la coordenada y que ocupa el nodo en el diagrama, sirve para actualizar la posición entre el modo nativo y transformado y al contrario. ❖ `private bool isVisited` Variable que indica si se ha pasado ya por un nodo al evaluar un diagrama de circuito eléctrico mediante el algoritmo de Tarjan.
Funcionalidades	Tiene funciones que no se han mejorado y se detallan en la memoria de Víctor Sacristán Tejudo.

Tabla 84: CLD-06: Node

5.2.2.2 Clases de interfaz

- **Clases Nuevas :**

CLD-07: NewArtifactTypeForm	
Responsabilidades	Interfaz para recoger los campos que son necesarios para crear un nuevo tipo de artefacto.
Atributos	❖ `private RepresentationType artifactType;` Objeto que almacena el nuevo tipo de artefacto que el usuario crea a través del formulario.
Funcionalidades	❖ `private void acceptButton_Click(object sender, EventArgs e)` Función que además de validar los diferentes campos que ha introducido el usuario en el formulario invocando las diferentes funciones de validación, envía al controlador el nuevo tipo de artefacto para que esté lo almacene en la base de datos. ❖ `private bool ValidationName(string text)` Función de validación del nombre del nuevo tipo de artefacto. ❖ `private bool ValidationAngle(int value)` Función que valida el ángulo introducido por el usuario. ❖ `private bool ValidationIcon(string text)` Función que valida la imagen que ha cargado el usuario (la imagen ya está en un string de Base 64) ❖ `private bool ValidationLayout(int value)` Función que valida que el layout que ha seleccionado el usuario es correcto. ❖ `private void addIconButton_Click(object sender, EventArgs e)` Función que despliega un menú para que el usuario pueda cargar una ventana del explorador, seleccione la imagen que necesite, y la asocie al nuevo tipo de artefacto que está creando. ❖ `public Bitmap resizeImage(Bitmap imgToResize, Size size)` Función que redimensiona la imagen que carga el usuario a un tamaño estándar con el que contarán en un principio todas las representaciones de los nodos en el diagrama cuando se transforme el mismo. ❖ `private void cancelButton_Click(object sender, EventArgs e)` Función que al pulsar el botón de cancelar paraliza la operación de creación de un nuevo tipo de artefacto.

Tabla 85: CLD-07: NewArtifactTypeForm

- **Mejoras sobre clases existentes:**

CLD-08: ArtifactForm	
Responsabilidades	Interfaz para recoger atributos que un usuario modifica de un artefacto.
Atributos	Tiene atributos que no se han mejorado y se detallan en la memoria de Víctor Sacristán Tejudo.
Funcionalidades	❖ `public Node checkFirstSelected(Node node)` Función que comprueba si se ha seleccionado el nodo como primero, así como también comprueba si ya hay otro nodo seleccionado como el primero en el diagrama. En dicho caso se despliega una ventana preguntando si se quiere seleccionar el que el usuario está editando en ese momento como el primero, o si lo prefiere, dejar el antiguo primero descartando los cambios. En ambos casos sólo queda un nodo en todo el diagrama como seleccionado. Está función sólo se carga cuando el diagrama tiene asociado un tipo de artefacto con una filosofía de dibujo eléctrico. Cuando se pulsa aceptar en la ventana el nodo que queda primero actualiza su atributo de isFirst marcándolo como el primero.

Tabla 86: CLD-08: ArtifactForm

CLD-09: TermForm	
Responsabilidades	Interfaz para recoger atributos que un usuario modifica de un término.
Atributos	Tiene atributos que no se han mejorado y se detallan en la memoria de Víctor Sacristán Tejudo.
Funcionalidades	❖ `public Node checkFirstSelected(Node node)` Función que comprueba si se ha seleccionado el nodo como primero, así como también comprueba si ya hay otro nodo seleccionado como el primero en el diagrama. En dicho caso se despliega una ventana preguntando si se quiere seleccionar el que el usuario está editando en ese momento como el primero, o si lo prefiere, dejar el antiguo primero descartando los cambios. En ambos casos sólo queda un nodo en todo el diagrama como seleccionado. Está función sólo se carga cuando el diagrama tiene asociado un tipo de artefacto con una filosofía de dibujo eléctrico. Cuando se pulsa aceptar en la ventana el nodo que queda primero actualiza su atributo de isFirst marcándolo como el primero.

Tabla 87: CLD-09: TermForm

CLD-10: TypeArtifactForm	
Responsabilidades	Interfaz para recoger el tipo de artefacto asignado al diagrama.
Atributos	❖ `private int representation;` Atributo que indica el tipo de filosofía de dibujo del tipo de artefacto seleccionado por el usuario de los almacenados en la base de datos.
Funcionalidades	❖ `public bool Init(Cake.Engine.CAKEEngine engine, Graph grafico)` Esta función es la que se encarga de cargar todos los datos desde la base de datos para incluirlos en el combo box que se muestra al usuario en la ventana. La mejora ha consistido en permitir cargar los tipos de artefacto con una filosofía de dibujo asociada, en vez de declararlos en el código como estaba hecho originalmente.

Tabla 88: CLD-10: TypeArtifactForm

5.2.2.3 Clases de control

- **Clases sobre las que se incorporan mejoras:**

CLD-11: graphicalArtifactUserControl	
Responsabilidades	Clase que contiene la lógica de negocio asignada al módulo de gestión de diagramas, tanto en el modo nativo como en transformado.
Atributos	Tiene atributos que no se han mejorado y se detallan en la memoria de Víctor Sacristán Tejudo.
Funcionalidades	❖ `public void newArtifactType()` Función que serializa en la base de datos los datos introducidos por el usuario en el formulario de creación de un nuevo tipo de artefacto. ❖ `internal void updateAllNodesInGraph()` Función que actualiza el diccionario que contiene todos los nodos del gráfico antes de aplicar un layout comprobando que se han guardado todos los cambios del usuario. ❖ `internal void updateLinkPoints(int idLink, string leftsToString, string topsToString)` Función para actualizar los puntos que se definen en las relaciones y así mantener los mismos entre el modo nativo y el modo transformado. ❖ `public void checkLayoutForArtifactTypes(Cake.Engine.CAKEEngine engine)`

	Función auxiliar que comprueba que están creados en la base de datos al menos los 4 tipos de artefactos asociados a los 4 tipos de filosofías de dibujo que se ofrecen en esté módulo, son: General, UML, UML-De secuencia y Circuito eléctrico. ❖ `public int checkFirstNode()` Función que recoge el primer nodo del diagrama, explícitamente seleccionado por el usuario, o el primero en añadirse al diagrama en su defecto. Usado para las funciones de aplicar layout de circuito eléctrico. ❖ `public void setDown(int down)` Función que actualiza con un atributo el tipo de caso de circuito eléctrico en el que se encuentra el diagrama. Consultar los casos en el punto de Configuraciones de circuitos eléctricos según las necesidades del cliente. ❖ `internal void updateFirstNode(int idFlow)` Función para marcar el nuevo primer nodo definido por el usuario. ❖ `internal void updateLastNode(int idFlow)` Función para marcar el último nodo en función del primer nodo que se haya definido en el diagrama. ❖ `internal void checkJustOneFirst(Node node, string key)` Función que comprueba que sólo haya un nodo definido como el primero en el diagrama, si hay varios selecciona el que recibe como parámetro. ❖ `internal void checkJustOneLast(Node node, string key)` Función que comprueba que sólo haya un nodo definido como el último en el diagrama, si hay varios selecciona el que recibe como parámetro. ❖ `internal void refreshDownCase()` Función que actualiza el caso de circuito eléctrico en el que se encuentra el diagrama cuando se va a aplicar la acción de layout de circuito eléctrico. ❖ `internal void tarjan()` Función que implementa el algoritmo de Tarjan. Este puede consultarse en el punto de Selección de algoritmos para detectar ciclos. ❖ `internal void strongConnect(Node v)` Función en la que se apoya la función de tarjan ( ) para aplicar los pasos de este algoritmo. ❖ `internal void invertRelation()` Función que define las relaciones que forman parte de un ciclo para que se traten adecuadamente al aplicar la acción de layout. ❖ `internal int searchFirstNode(List<Node> listNodesInCycle)`

	Función para encontrar y marcar el primer nodo de un ciclo. ❖ `internal bool setCircuitProperties()` Función para recoger en cada nodo las propiedades necesarias para poderlos evaluarlos uno a uno en el algoritmo de tarjan ( ). ❖ `internal Node getFirstNodeSCC(List<Node> list, int first)` Función que devuelve el primer nodo de una componente fuertemente conexa. ❖ `internal int getFirstInCycle()` Función que devuelve el identificador de la primera relación de un ciclo. ❖ `internal int getLastInCycle()` Función que devuelve el identificador de la última relación de un ciclo. ❖ `internal void relationsToOriginalState()` Función para deshacer acciones realizadas sobre las relaciones después de aplicar el algoritmo de tarjan. ❖ `internal bool isNodeTurned(Node node)` Función para deshacer acciones realizadas sobre los nodos afectados por relaciones, después de aplicar el algoritmo de tarjan.

Tabla 89: CLD-11: graphicalArtifactUserControl

CLD-12: Transformation	
Responsabilidades	Clase que contiene la lógica de negocio asignada al módulo de transformación de diagramas.
Atributos	Tiene atributos que no se han mejorado y se detallan en la memoria de Víctor Sacristán Tejudo.
Funcionalidades	❖ `public string canTransformation()` Función que comprueba el tipo de artefacto asociado al dibujo, ya que sólo se permiten representar diagramas asociados a tipos de artefactos con las siguientes filosofías de dibujo: circuito eléctrico, estructural y UML-De Secuencia. En caso de no cumplirse esta condición se mostrará un mensaje efímero mostrando que no se puede realizar la transformación al usuario. Devuelve una clave para saber que transformaciones usar, UML o Secuencia originales, o la nueva implementación en función de la filosofía de dibujo asociada al diagrama. ❖ `public void drawTransformationCustom()` Función para poder realizar las nuevas transformaciones con imágenes asociadas a las representaciones de los nodos. Esta función recorre todos los nodos del diagrama evaluando si se deben rotar o no en función de la posición que ocupan en el diagrama y de la filosofía de dibujo que tiene asociada el mismo. Una vez llama a todas las funciones para rotación de imágenes crea un objeto que contiene al nodo para mandarlo a la librería gráfica en JavaScript. Además también trata todos los términos y las relaciones, convirtiéndolos en objetos que manda a JavaScript para que sean representados también en el modo transformado como estaban originalmente en el modo nativo. ❖ `internal double Angle(double p1x, double p1y, double p2x, double p2y)` Función que calcula el ángulo que forman los puntos por los que se conectan una relación entrante y una saliente a un nodo. ❖ `internal bool checkPointWithInLine(double pIniX, double pIniY, double pFinX, double pFinY, double pointX, double pointY)` Función para comprobar que las coordenadas del centro de un nodo están en línea con los puntos de entrada de alguna de las relaciones entrantes al mismo, así como en línea con el punto definido por alguna de las relaciones salientes del nodo. ❖ `internal double rotateNodeBitmap(Node nodoAux)`

	Función que rota la representación asociada a un nodo antes de representarse en la ventana de transformación, realiza la rotación de la misma basándose en las dos funciones inmediatamente anteriores, teniendo en cuenta diferentes casos en los que la imagen debe rotar. ❖ `internal double getNRelationCases(List<Relation> relationsIn, List<Relation> relationsOut, double centerPointNodeX, double centerPointNodeY, int caso)` Función que realiza la misma operación que rotateNodeBitmap ( ), sólo que en los casos en los que haya varias relaciones entrantes al nodo y varias salientes (es decir en números de 1: N, N: N y N: 1). Así calcula desde qué relaciones entrantes y salientes, teniendo en cuenta el punto central que define las coordenadas que ocupa un nodo, se forma un eje en el que pueda rotar la imagen, siguiendo un orden: Primero los casos que formen un eje horizontal, luego los casos que formen un eje vertical y por último los casos en los que se forme un eje dando igual el ángulo. ❖ `internal double getResultRelationOut(List<Relation> relationsOut, double centerPointNodeX, double centerPointNodeY)` Función que al igual que las anteriores rota una imagen de un nodo en función del eje que formen sus relaciones con el centro del mismo, sólo que en este caso se evalúa el caso en el que existan varias relaciones salientes pero ninguna entrante, en este caso también se seguirá el orden de prioridad definido para el método anterior según el ángulo que formen los ejes. ❖ `internal double getResultRelationIn(List<Relation> relationsIn, double centerPointNodeX, double centerPointNodeY)` Función que al igual que las anteriores rota una imagen de un nodo en función del eje que formen sus relaciones con el centro del mismo, sólo que en este caso se evalúa el caso en el que existan varias relaciones entrantes pero ninguna saliente, en este caso también se seguirá el orden de prioridad definido para el método anterior según el ángulo que formen los ejes. ❖ `internal double calculateNearestValue(List<double> angles, double searchValue)` Función que se usa para ordenar una lista de ángulos que definen los ejes relación-nodo-relación, de tal manera que los pondere desde los más cercanos a 0 º o 180 º. (En realidad todas estas funciones trabajan en radianes debido a que .NET proporciona una API mucho más extendida para operaciones con radianes que para operaciones con grados).

Tabla 90: CLD-12: Transformation

Cabe por último destacar que la funcionalidad de la clase Transformation se ha incluido en una región de graphicalArtifactUserControl para disminuir la complejidad entre las llamadas y la librería gráfica y usar la

interfaz ScriptManager en la medida de lo posible, evitando que la librería se comunique directamente con graphicalArtifactUserControl.

5.2.3 Especificación de mejoras sobre los scripts

En este apartado se van a detallar las mejoras realizadas sobre los scripts de la librería gráfica creada en la etapa anterior de desarrollo. Cabe destacar que no ha sido necesario crear ninguna clase nueva, y por tanto sólo se han añadido nuevas funcionalidades al código presente en las mismas.

Si se atiende al diagrama incluido en la figura 30 incluida en el apartado de Definición de niveles de arquitectura, se podrán observar todas las clases que se han desarrollado, o mejorado en la librería gráfica para cumplir con los objetivos estipulados el año pasado. Toda esta información viene detallada y justificada en el punto 5.1.2 Diseño del conjunto de responsabilidades de los scripts de la memoria de Pablo Sánchez Pérez. Así este apartado sólo se centrará en documentar las mejoras implementadas para poder cumplir con los requisitos de esta etapa de desarrollo para el sistema.

Debido a la extensión de la librería (algunas clases cuentan con más de 8000 líneas de código), ya sólo entender toda la funcionalidad implementada en la misma ha sido una tarea con un esfuerzo exageradamente alto. Esto junto con el poco detalle en comentarios de código ha hecho que sea una de las tareas que más esfuerzo y tiempo ha costado en el desarrollo de este proyecto (sino la que más).

Otro factor limitante consiste en la implementación con la que cuenta la librería, ya que está pensada para usarse directamente, y claramente no es fácil modificarla para cumplir con las necesidades que se requieren en cualquier proyecto.

A continuación se detallarán las diferentes clases que se han modificado siguiendo con el esquema y la plantilla usada en el punto de Especificación de clases. Se han cambiado las siglas de identificación por CLJ (J de JavaScript), además se excluirán los atributos de esta especificación de clases ya que son implementaciones internas de la librería que se detallan también a lo largo de la memoria de Pablo Sánchez Pérez, y no tienen valor añadido para lograr comprender las mejoras implementadas y los métodos que se van a explicar a continuación (ya que estos se basan en las entidades que se crearon el año pasado y que se usan en toda esta mejora, nodo, relación… Siendo esta parte superficial la que se debe comprender para entender esta nueva funcionalidad).

La principal clase que se ha modificado será layoutflow, en la cual se implementará toda la lógica relacionada con la colocación de los elementos en diagramas de circuito eléctrico, siendo además una de las clases más extensas de todo el proyecto sólo por detrás de la propia clase que sirve como base para esta librería, la clase addflow, con 6503 líneas de código.

CLJ-01: addflow	
Responsabilidades	Librería que efectúa toda la gestión gráfica de los diferentes diagramas mostrados en el canvas contenido en todas las ventanas en las que se visualizan diagramas
Funcionalidades	❖ `function initLinkPointsPolyline(_link)` Función que se encarga de pintar las relaciones reflexivas, se ha modificado para que pinte estás relaciones en forma cuadrada, de tal manera que se siga la convención establecida por el cliente para circuitos eléctricos. ❖ `function resizeImageNode(_node)` Función que se encarga de redimensionar las imágenes según el usuario modifica el tamaño de algún nodo, de tal manera que la imagen se ajusta al mismo.

Tabla 91: CLJ-01: addflow

CLJ-02: layoutflow	
Responsabilidades	Librería que actuando junto a addflow se encarga de realizar la recolocación automática de los elementos presentes en un diagrama según la filosofía de dibujo que tiene asociada (árbol, circuito eléctrico…)
Funcionalidades	Para localizar todas estas mejoras, estas se encuentran dentro de la implementación para los circuitos eléctricos dentro de esta función (en JavaScript existen funciones declaradas dentro de funciones sin problema): • `SeriesParallel: function (flow, layerDistance, vertexDistance, drawingStyle, orientation, xvertex, yvertex, xmargin, ymargin)` Dentro de esta función es donde se implementarán todas las mejoras relacionadas con el diagrama de circuito eléctrico. Está función no era usada en el desarrollo realizado en la etapa anterior de este sistema, así que ha sido modificada e incorporada al sistema siendo sus funciones principales: • `function flowToGraph()` Función que recoge los nodos y relaciones del diagrama y los guarda en una estructura de datos propia de la librería basada en vértices y aristas propias de un grafo direccional, de tal manera que así aplica los distintos algoritmos. Además como modificaciones implementadas sobre esta función, se permite marcar el primer y último nodo, de manera que según su posición en el dibujo, se pueden ordenar los nodos según el camino que define el circuito desde el primer nodo al último, ordenándolos por en orden creciente en el eje x del diagrama mediante la nueva función implementada: `function sortfunction(a, b)`

	Por último evalúa el primer y el último nodo para saber si tiene que recalcular las coordenadas donde están situados bajándolos como se ha definido en el punto de Configuraciones de circuitos eléctricos según las necesidades del cliente. • `function addVertex(_node)` Función que convierte un nodo a la estructura de datos (vértices) usada para esta funcionalidad. • `function addEdge(org, dst, _link)` Función que convierte una relación a la estructura de datos (vértices) usada para esta funcionalidad. • `function delVertex(v)` Función que elimina un vértice de los almacenados para el tratamiento de un grafo direccional. • `function delEdge(e)` Función que elimina una arista de las almacenadas para el tratamiento de un grafo direccional. • `function getEdge(org, dst)` Función que permite recoger una arista de las almacenadas para el grafo direccional si se define su origen y su destino (el nodo del que parte, y el nodo al que se dirige). • `function graphToFlow(params)` Función que recalcula la posición que deben seguir los vértices y aristas del grafo direccionado que tiene que tener forma de circuito eléctrico. Es una de las funciones más importantes de esta clase ya que permite aplicar el algoritmo de dibujo basado en serie-paralelo para recalcular la posición de todos los elementos del diagrama, devolviendo nodos y relaciones desde los vértices y aristas del grafo direccional, que se pueden por fin pintar en el canvas formando la nueva disposición del diagrama. Además es aquí donde a partir de las nuevas posiciones que ocupan los nodos del diagrama (establecidas en flowToGraph ( )), se calculan los puntos que definen a las nuevas relaciones para lograr los resultados esperados por el cliente como de nuevo se muestra en el punto Configuraciones de circuitos eléctricos según las necesidades del cliente. Además es en esta función donde se empiezan a evaluar los primeros ciclos. Estos se han detectado anteriormente desde la capa de control mediante el algoritmo de tarjan y aquí llegan para poder tratarse y pintarse correctamente. Esto se hace llamando a la función: • `function isCycleDirected()` Está función recoge todos los ciclos detectados en el diagrama para tratarlos adecuadamente y establecer los casos que debe tratar de dibujar graphToFlow (params). • `function SPLayout()`

	Función que aplica el algoritmo de series-paralelo especificado en el punto de Proceso de selección de algoritmos para dibujar diagramas de circuitos eléctricos.

Tabla 92: CLJ-02: layoutflow

CLJ-03: customShapes	
Responsabilidades	Script en la que se permite implementar las formas nuevas necesarias para la librería
Funcionalidades	• `custom.nodeCustomClass = function (titleName, attributes, methods, idFlow, typeNode)` Función que actúa como una clase dentro de customShapes y que recoge las nuevas formas que contendrán los nodos con tipos de artefactos que no tienen asociadas cierta representación, pero se encuentran en un diagrama que si tiene elementos con representación, por lo que muestra estos en modo nativo por especificación del cliente como se muestra a continuación:  *Ilustración 34: Ejemplo de forma personalizada de un elemento sin representación gráfica en un diagrama que sí cuenta con elementos con representaciones* • `custom.nodeCustomClassImage = function (titleName, image, idFlow, typeNode, direction, width, height)` Función que actúa como una clase dentro de customShapes y que recoge las nuevas formas que contendrán imágenes para los nodos con tipos de artefactos, y que tienen asociadas cierta representación, cargando la misma una vez se crea la forma en el diagrama. Todas estas imágenes contarán con una representación en el diagrama del mismo tamaño para todos los nodos, pero gracias a la nueva funcionalidad implementada para la clase addflow, se permite redimensionarlas según las necesidades del usuario. Un ejemplo se muestra a continuación:  *Ilustración 35: Ejemplo de forma personalizada de un elemento con representación gráfica* También se establece la posición del texto asociado al nodo en función de la configuración que ha adquirido el dibujo, esto se aprecia perfectamente en la ilustración 10 en el punto de Circuitos en serie/paralelo cerrados.

Tabla 93: CLJ-03: customShapes

CLJ-04: manager	
Responsabilidades	Script que permite la comunicación de la librería gráfica con otros subsistemas.
Funcionalidades	❖ `function createRelation(relation, secuencia)` Función que crea una relación para que la librería gráfica la procese recibiendo estos datos desde la capa de controlador como una entidad de tipo Relation que ha convertido en un objeto accesible desde JavaScript. Como mejora, se ha implementado la actualización de los puntos de la relación cuando se dibuja la misma también en la ventana de transformación. De tal manera se guardan las coordenadas que definen todos los puntos de una relación para poder replicar su configuración desde el modo nativo al transformado y al contrario. ❖ `function createNodeCustom(nodeManager, GeneralProperties)` Función nueva que permite en diagramas con nodos que tienen asociada una representación, pintar cada nodo, teniendo en cuenta si tienen o no representación. Se usa en el modo transformado ya que es el único en el que se pueden pintar dichas representaciones. Esto lo hacen llamando a las nuevas funciones implementadas para el script customShapes. ❖ `function updateNode(updateNode, updatePosition)` Función modificada que actualiza un nodo en el canvas cuando se ha editado, ya sean sus propiedades, o ahora también la posición que ocupa y sus dimensiones. Quedando estás actualizadas para cumplir con la necesidad de mantener el mismo esquema entre el modo nativo y el transformado y al contrario. ❖ `function hideLinks()` Función que oculta en la paleta de edición, todo los tipos de relaciones, excepto las relaciones de asociación, que se pueden añadir a un diagrama en el modo nativo si el diagrama seleccionado es de circuito eléctrico. ❖ `function showLinks()` Función que muestra en la paleta de edición todos los tipos de relaciones que se pueden añadir al canvas siempre y cuando el diagrama no sea de circuito eléctrico. ❖ `function invertRelation(relation)` Función que permite dibujar las relaciones pertenecientes a componentes fuertemente conexas en circuitos eléctricos, y que es usada para completar el algoritmo de Tarjan.

Tabla 94: CLJ-04: manager

CLJ-05: toolBar	
Responsabilidades	Script que contiene todas las funciones recogidas en la barra de herramientas.
Funcionalidades	❖ `function layout()` Función modificada para aplicar distintos layouts dependiendo de la filosofía de dibujo que tenga asociada un diagrama, de tal manera que indiferentemente de la misma se aplique la acción de layout. Además se ha incluido la llamada a la acción de layout de circuito eléctrico. ❖ `function layoutCircuit()` Función que invoca la acción de layout de circuito eléctrico para los elementos de un diagrama aplicando el nuevo layout series/paralelo definido en layoutflow. ❖ `function changeVerbose()` Función modificada para poder permitir el verbose en tipos de diagrama que cuenten con representación (denominado modo general en todo el sistema), de tal manera, se habilitará la acción de verbose en los elementos del diagrama que no cuenten con representación, siguiendo el mismo modelo definido para el modo nativo. Está función también se aplica en el modo nativo, pero no se apreciará está modificación hasta que se aplique el verbose en el modo transformado. ❖ `function newArtifactType()` Función nueva que permite cargar la acción para crear un nuevo tipo de artefacto. ❖ `function changeDiagramMode()` Función modificada que se comunica con la capa controladora para comenzar con el proceso de transformación del diagrama cuando tiene asociada la nueva filosofía de dibujo de circuito eléctrico o cuenta con nodos con representación. ❖ `function updateAllPositions()` Función que se encarga de actualizar las posiciones de los nodos entre el modo nativo y el transformado y al contrario enviándolas a la capa de control. Esta función ha sido modificada para guardar también los puntos que definen las relaciones, así como el ancho y el alto de los nodos para preservar que el diagrama sea el mismo entre el modo nativo y el transformado.

Tabla 95: CLJ-05: toolBar

5.3 Diseño de casos de uso reales

En este apartado se elaborarán diagramas de secuencia para cada uno de los casos de uso definidos en la parte de análisis de este documento.

Como ya se explicó en ese punto, los diagramas de secuencia se van a especificar en el apartado de diseño puesto que cuentan con mucho más detalle que si se hubieran hecho en la parte de análisis. Esto es gracias a que aquí se definen las funciones reales que se han diseñado para poder cumplir con cada caso de uso definido.

Al contrario que en el año pasado, en el que los desarrolladores se centraban en una capa de la arquitectura considerando el resto como una caja negra, en este proyecto se mostrarán diagramas de secuencia indicando la comunicación entre diferentes capas de la arquitectura mediante las llamadas que tienen entre ellas. Se ha suprimido la interfaz ScriptManager, ya que su función no es más que llamar a funciones contenidas en graphicalArtifactUserControl. Además y como se especificó en el apartado de clase Clases de control, la clase Transformation ha sido integrada dentro de graphicalArtifactUserControl, debido principalmente hay que existían numerosas funciones duplicadas y la existencia de la clase no tenía mucho sentido debido a la gran dependencia de la clase graphicalArtifactUserControl (la mayoría de los métodos que actualizan la propia ventana de transformación, son los mismos usados para mostrar el modo nativo en este sistema).

A continuación se especificarán todos los diagramas de secuencia que hacen referencia a cada caso de uso definido en el apartado de análisis:

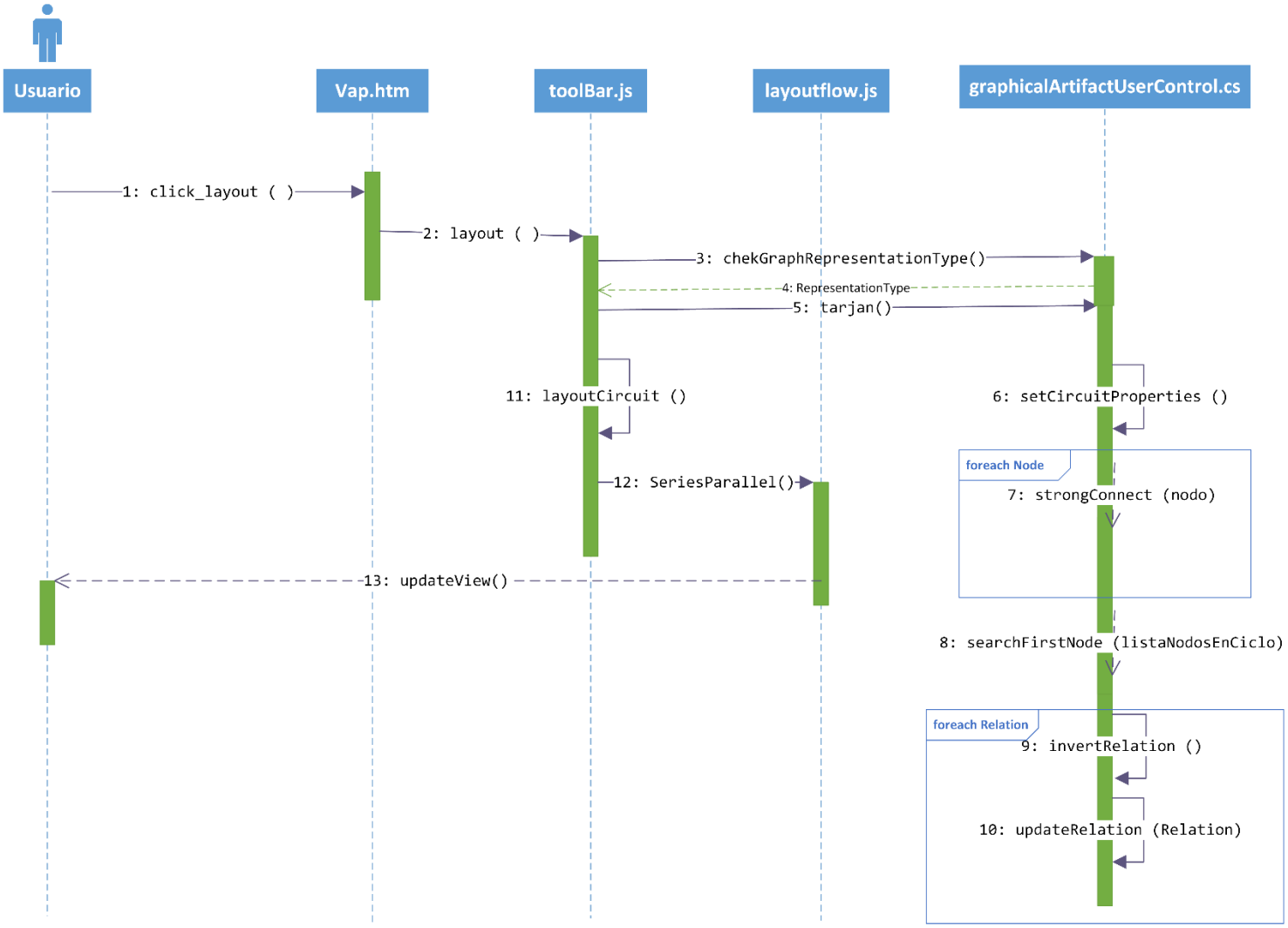

Ilustración 36: Diagrama de secuencia CU-001: Efectuar acción de layout, para filosofía de dibujo asociada a un diagrama de circuito eléctrico en el modo nativo

Una aclaración importante para el siguiente diagrama de secuencia, es que va a integrar los casos de uso CU-02 y CU-03, que son los que hacen referencia a dos casos de layouts en el modo transformado.

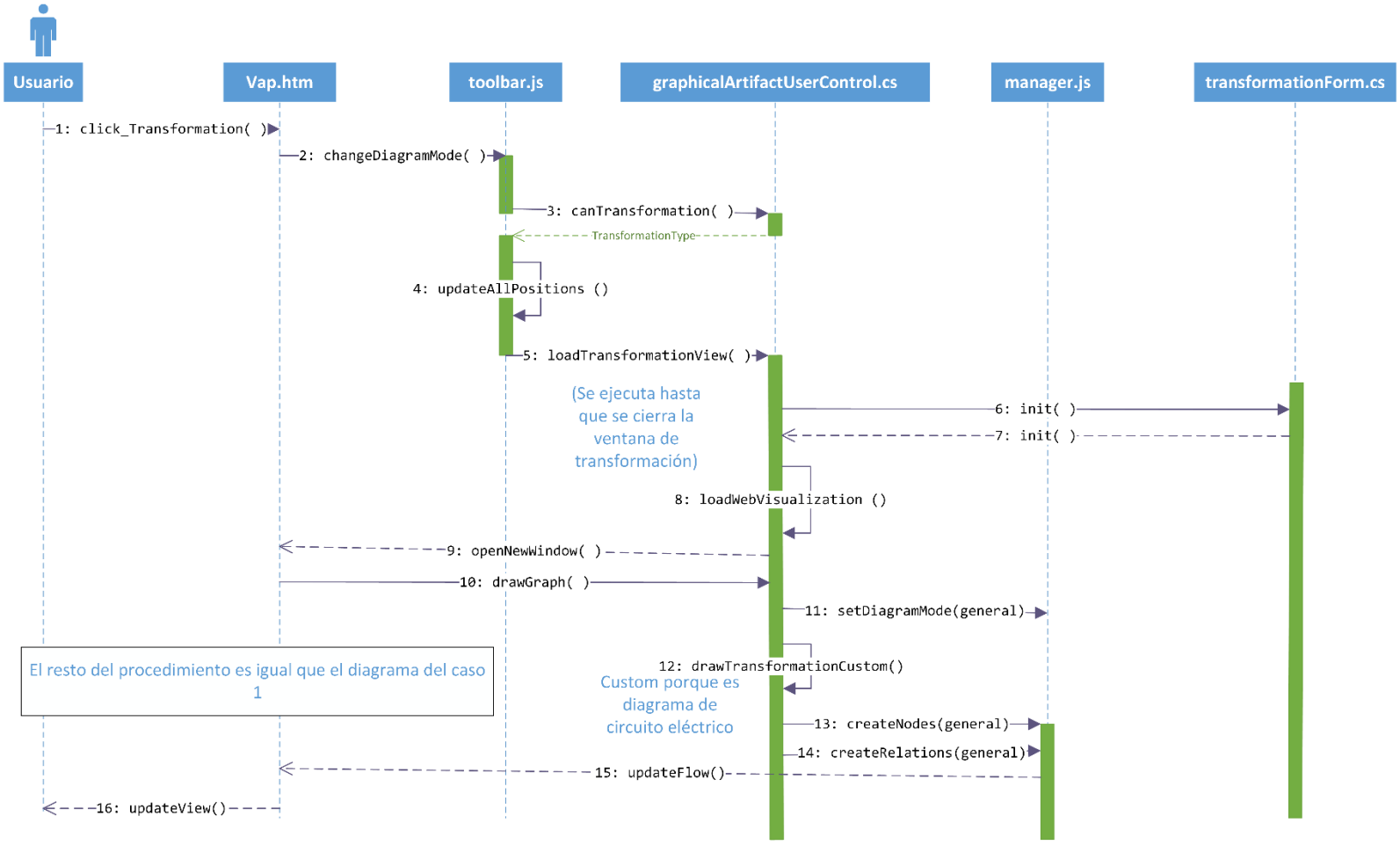

Ilustración 37: Diagrama de secuencia CU-002: Efectuar acción de layout, para filosofía de dibujo asociada a un diagrama de circuito eléctrico en el modo transformado.

Partiendo del diagrama anterior se puede definir también el caso de layout de UML en la ventana de transformación para mostrar que dicha funcionalidad está implementada para los layouts antiguos. Por lo tanto el siguiente diagrama continuaría justo después de donde terminar el diagrama anterior para el caso de layout UML en el modo transformado.

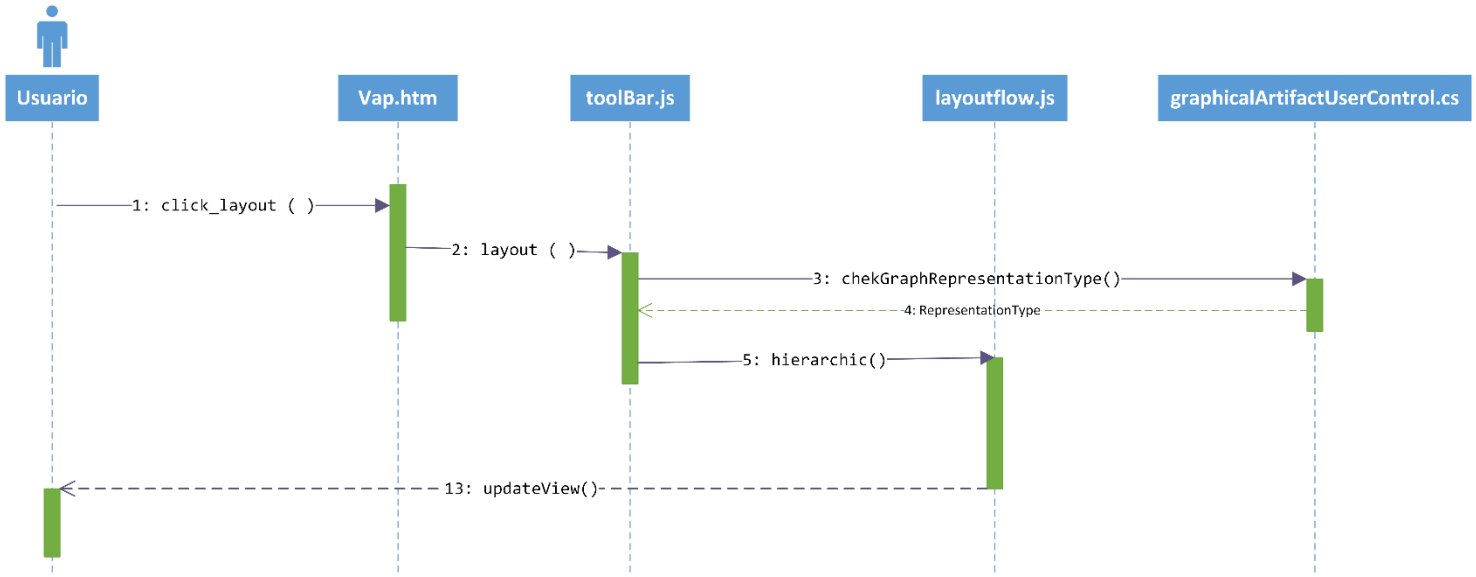

Ilustración 38: Diagrama de secuencia CU-003: Efectuar acción de layout, para filosofía de dibujo asociada a un diagrama UML en el modo transformado

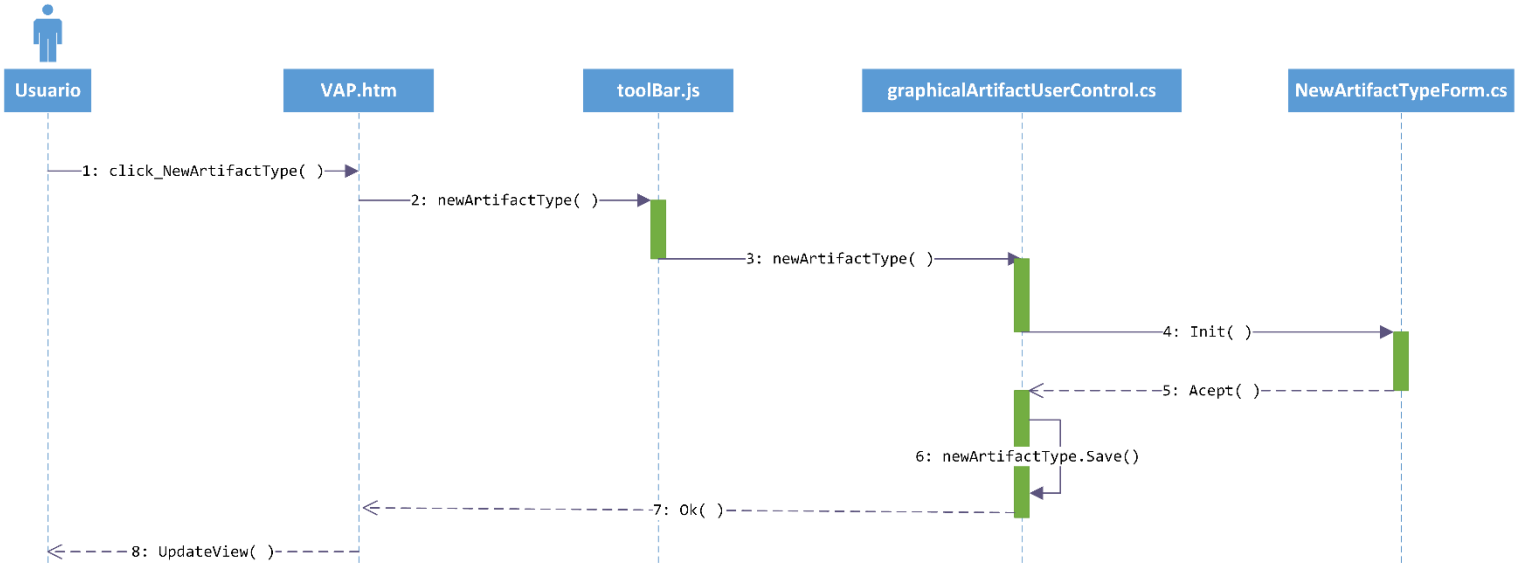

Ilustración 39: Diagrama de secuencia CU-004: Añadir nuevos tipos de artefactos

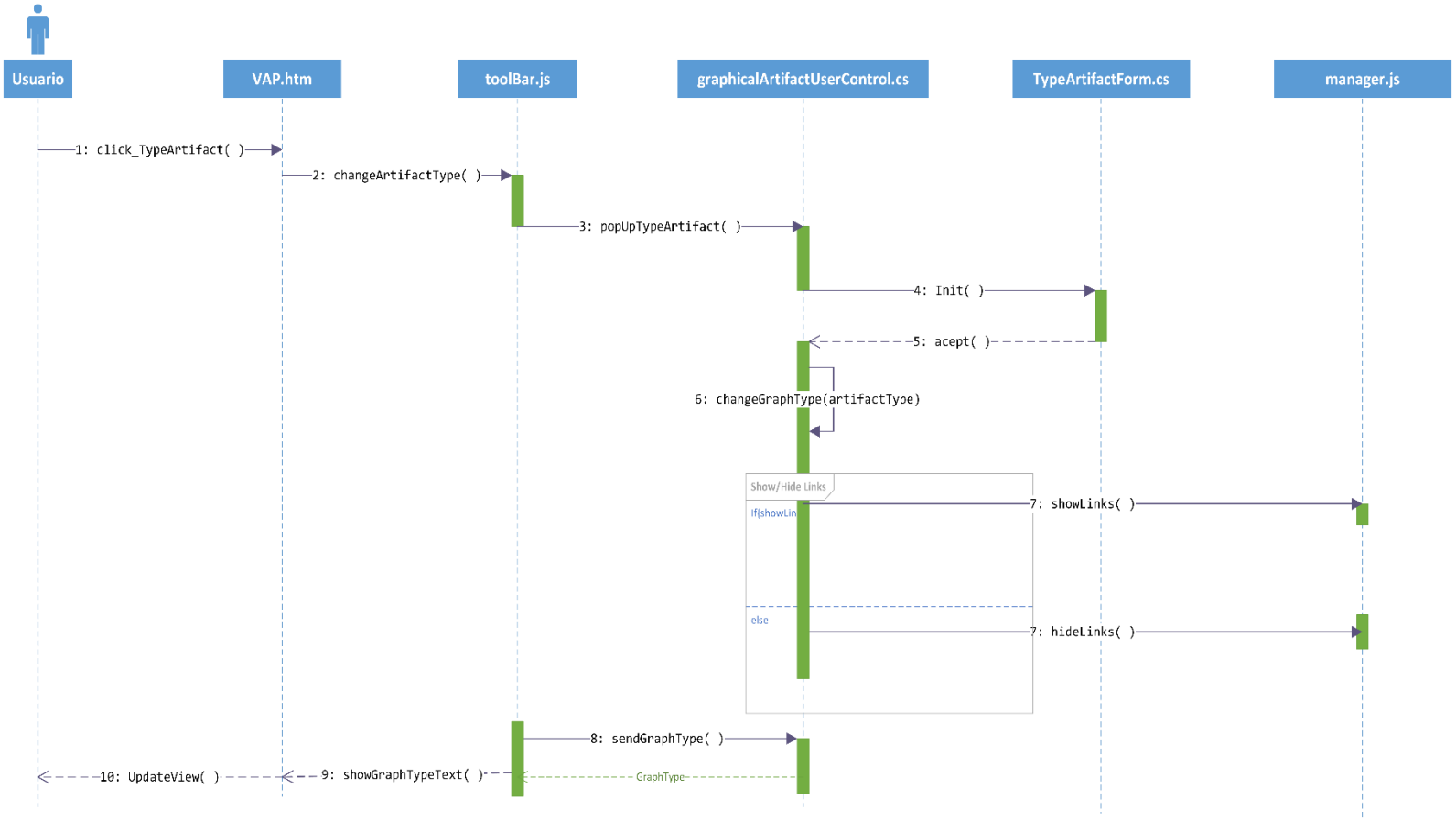

Ilustración 40: Diagrama de secuencia CU-005: Definir layout del diagrama

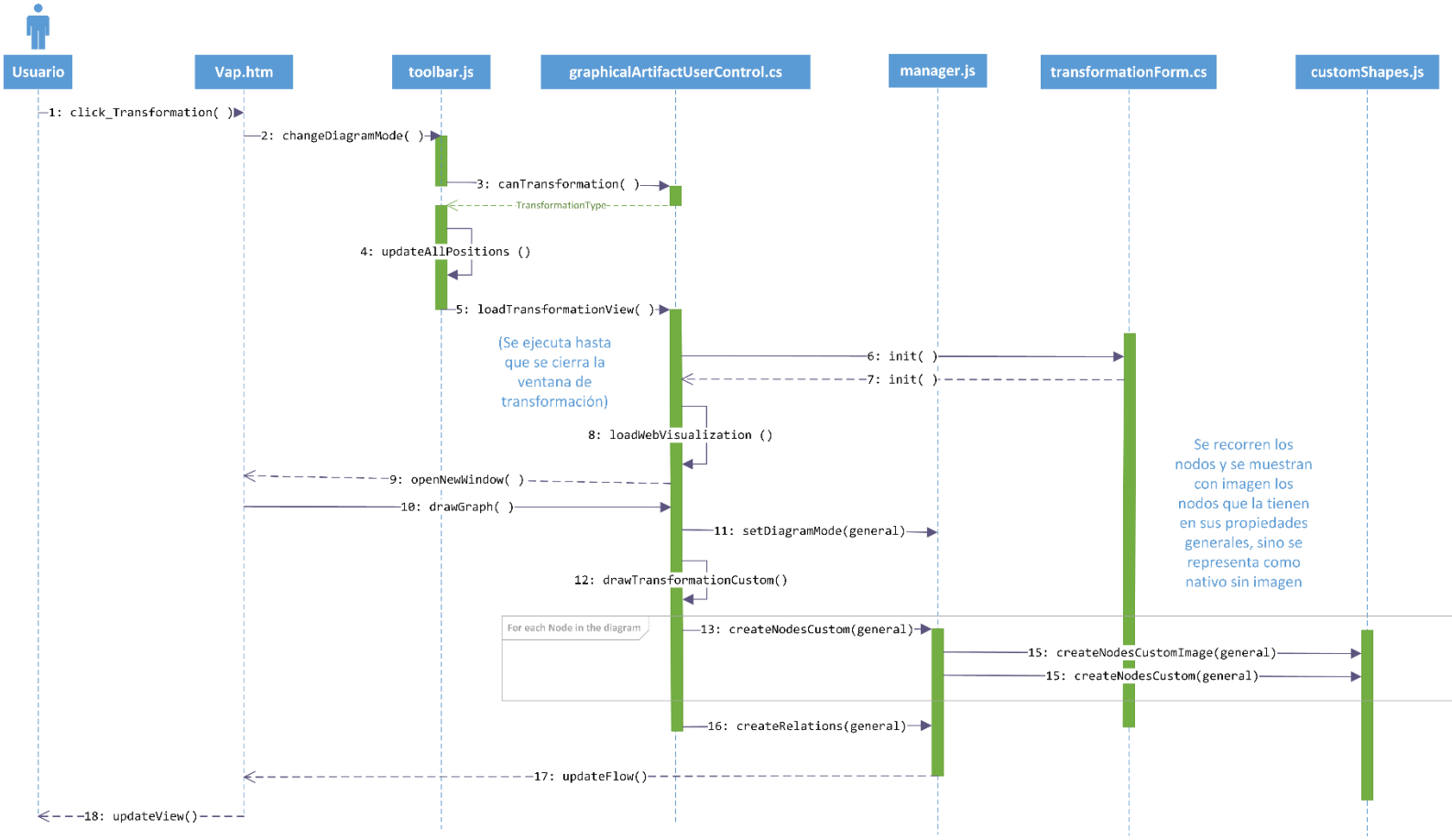

Ilustración 41: Diagrama de secuencia CU-006: Transformar el diagrama con elementos que tienen asociada una representación

A continuación en el caso de usuario 7, se mostrará con más detalle cómo se actualizan las posiciones de los nodos y de las relaciones en el método changeDiagramMode( ), especificando sólo esta parte ya que el resto del diagrama sería igual a los anteriores que hacen referencia al modo transformado.

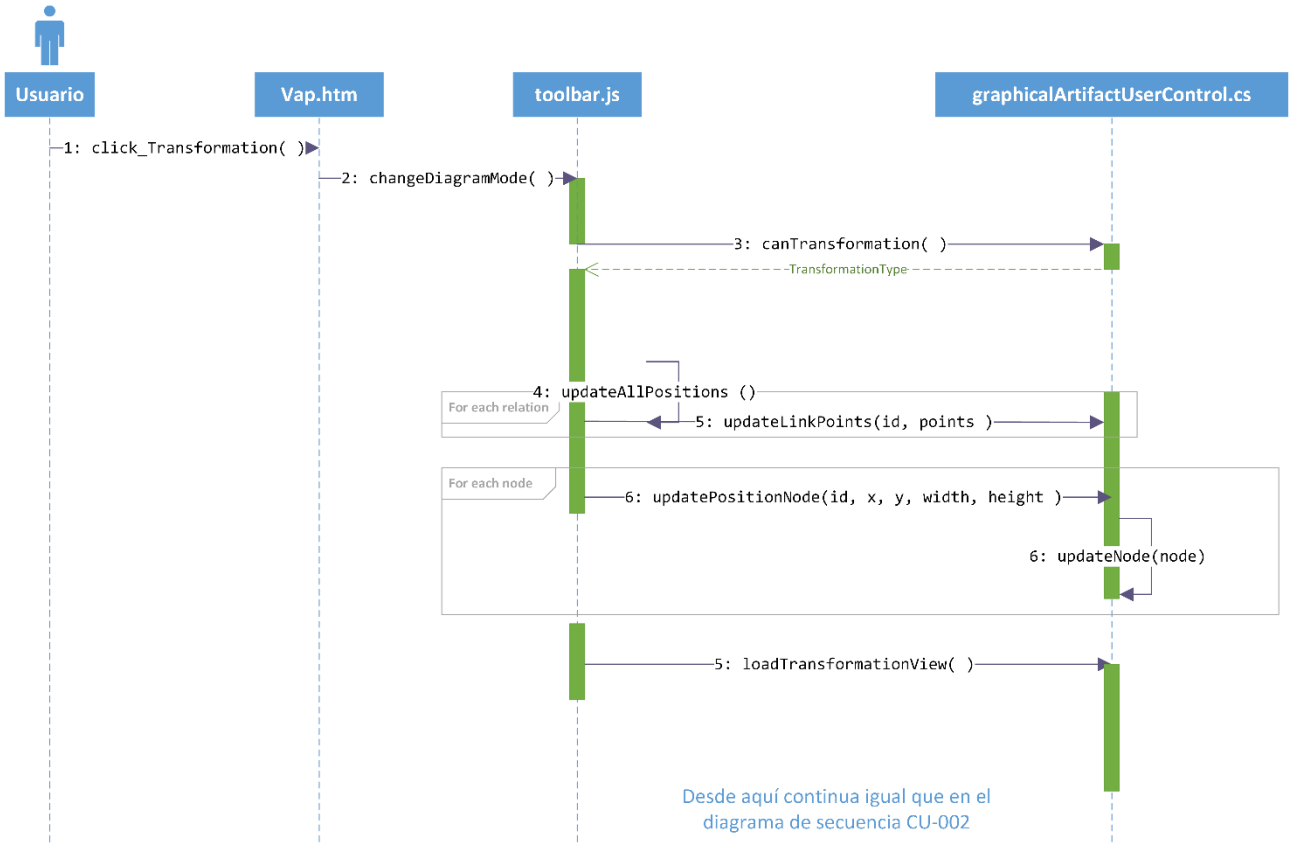

Ilustración 42: Diagrama de secuencia CU-007: Recolocar los elementos del diagrama

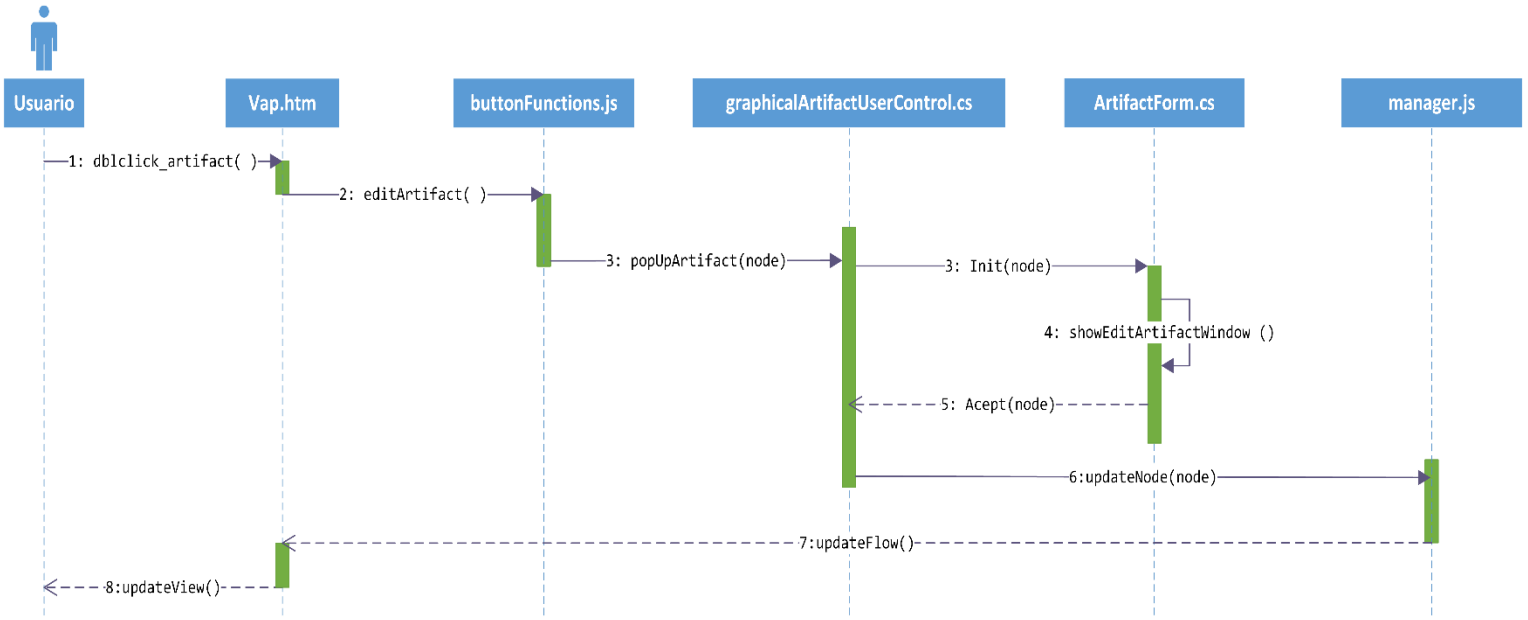

Ilustración 43: Diagrama de secuencia CU-008: Modificar tipo de un artefacto

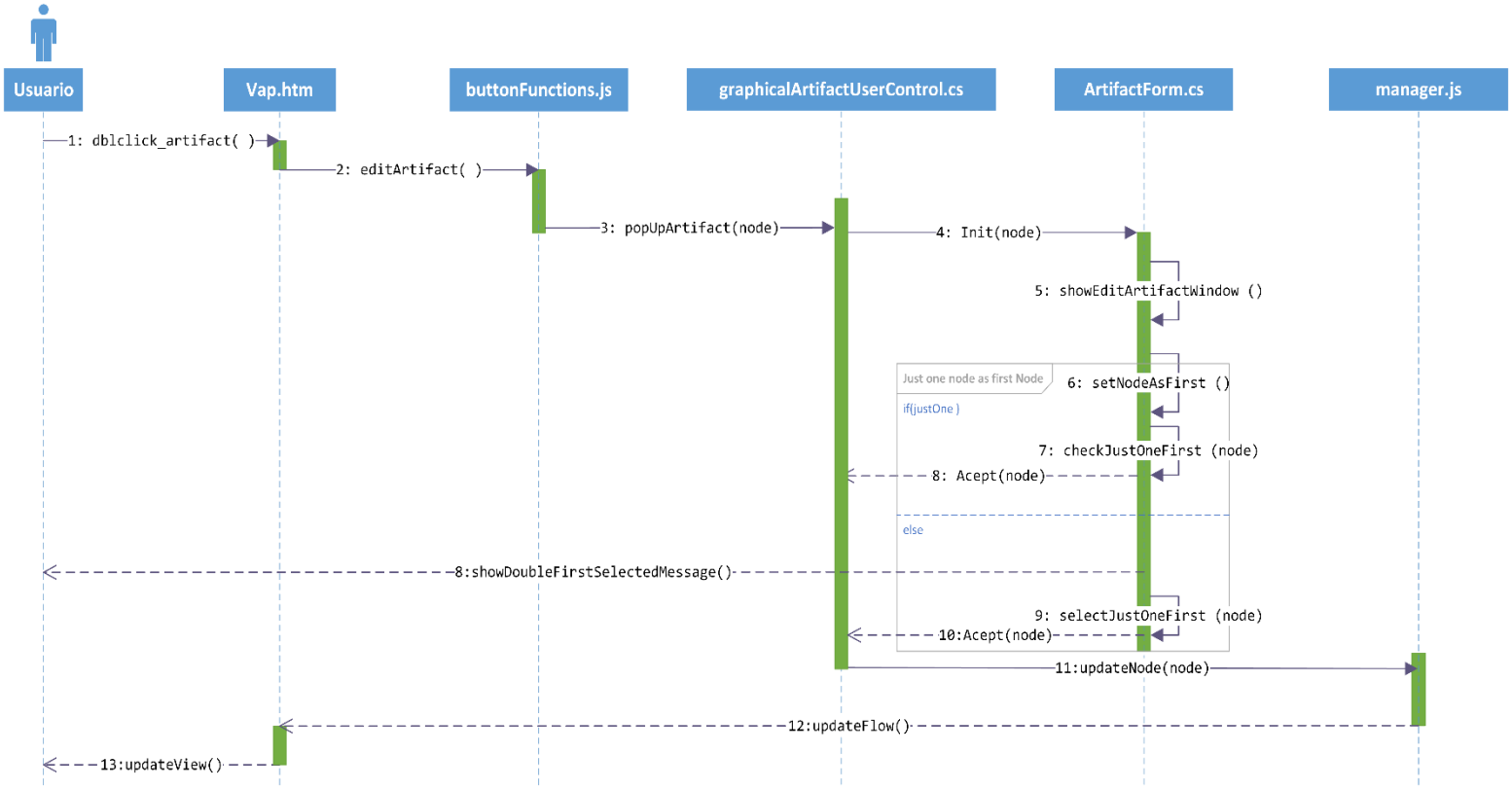

Ilustración 44: Diagrama de secuencia CU-009: Seleccionar primer elemento de un ciclo en circuitos eléctricos

5.4 Especificación técnica del plan de pruebas

Siguiendo la arquitectura definida anteriormente, la lógica de negocio se encuentra en la capa de control. Así la función de este apartado es recoger las diferentes pruebas para comprobar que esta capa funcione siempre correctamente durante el desarrollo de la mejora sobre este sistema.

Es importante destacar que en este punto sólo se van a recoger las distintas pruebas realizadas sobre las mejoras implementadas en este módulo, aunque se tienen en cuenta todas y cada una de las pruebas definidas en la etapa de desarrollo anterior de este proyecto, las cuáles se pueden consultar, como se ha repetido en varias ocasiones a lo largo de este documento, en las memorias de los distintos desarrolladores.

Para estas pruebas y como se especificó en el punto de Gestión de pruebas, se empleará "Nunit" como software de control de pruebas unitarias para la plataforma .NET.

5.4.1 Definición del alcance de las pruebas

Para el desarrollo de este sistema se estableció en el apartado de Análisis un plan de pruebas de aceptación, el cuál cubre prácticamente todas las necesidades de las nuevas mejoras implementadas en este proyecto de fin de grado. Esto se debe principalmente a que casi todas las pruebas están orientadas a pruebas de UI (User interface – Interfaz de Usuario), ya que todas las nuevas funcionalidades añadidas a este sistema se centran es elementos visuales que son muy difíciles de comprobar de otra forma.

Aun así, y junto a todas estas pruebas de aceptación, se han creado pruebas unitarias enfocadas a comprobar las principales clases de control de este sistema.

5.4.2 Pruebas unitarias

Para recoger todas estas pruebas se usará la siguiente plantilla, siendo **PU** las siglas de prueba unitaria.

PU- ## : Nombre de la prueba unitaria	
Descripción	Descripción del cometido que cumple esta prueba unitaria.
Función	Función a la que hace referencia esta prueba unitaria y se quiere probar.
Especificación de entrada	Elementos a instanciar para poder comprobar la prueba.
Especificación de salida	Valor esperado para que la prueba unitaria se considere exitosa

Tabla 96: Plantilla para la tabla de pruebas unitarias del sistema

A continuación se detallarán estas pruebas unitarias clasificadas por la clase que contiene las funciones que están comprobando.

- **Pruebas unitarias sobre graphicalArtifactUserControl:**

PU- 01: Comprobar nueva transformación con nodos con representación	
Descripción	Se comprueba que el diagrama con nodos que tienen asociada alguna representación se puede transformar mostrando las imágenes en la ventana de transformación.
Función	`canTransformationCustom()`
Especificación de entrada	Crear un gráfico con nodos que sean artefactos y que tengan asociada alguna representación.
Especificación de salida	Se devolverá un booleano indicando si se puede o no transformar el diagrama con imágenes asociadas a los nodos.

Tabla 97: PU- 01: Comprobar nueva transformación con nodos con representación

PU- 02: Comprobar primer nodo de diagramas de circuito eléctrico	
Descripción	Se desea probar la función que define cuál es el primer nodo en un gráfico de circuito eléctrico.
Función	`checkFirstNode()`
Especificación de entrada	Crear un gráfico. Seleccionar para este gráfico layout de circuito eléctrico. Añadir varios nodos conectados por relaciones.
Especificación de salida	Tras la llamada a esta función se devuelve un entero que hace referencia al identificador de un nodo concreto siendo este el primero del circuito. Automáticamente se selecciona como primer nodo de un circuito eléctrico el primero que se añade al mismo, sino se ha editado un nodo por parte del usuario seleccionándolo como primer elemento.

Tabla 98: PU- 02: Comprobar primer nodo de diagramas de circuito eléctrico

PU- 03: Comprobar primer nodo de una componente fuertemente conexa.	
Descripción	Se desea probar la función que define cuál es el primer nodo en un gráfico de circuito eléctrico.
Función	`searchFirstNode()`

Especificación de entrada	Crear un gráfico. Seleccionar para este gráfico layout de circuito eléctrico. Añadir varios nodos conectados por relaciones formando algunas retroalimentaciones en el circuito eléctrico. Aplicar la acción de layout.
Especificación de salida	Tras la llamada a esta función se devuelve un entero que hace referencia al identificador de un nodo concreto siendo este el primero de la componente fuertemente conexa.

Tabla 99: PU- 03: Comprobar primer nodo de una componente fuertemente conexa.

PU- 04: Comprobar que se guardan en los nodos la propiedades necesarias para diagramas de circuitos eléctricos	
Descripción	Se comprueba que se guardan en los nodos la propiedades necesarias para aplicar el algoritmo de Tarjan en diagramas de circuitos eléctricos con ciclos.
Función	`setCircuitProperties()`
Especificación de entrada	Crear un gráfico. Seleccionar para este gráfico layout de circuito eléctrico. Añadir varios nodos conectados por relaciones. Aplicar la acción de layout.
Especificación de salida	Tras la llamada a esta función se devuelve un booleano indicando que se han podido cargar las propiedades en los nodos.

Tabla 100: PU- 04: Comprobar que se guardan en los nodos la propiedades necesarias para diagramas de circuitos eléctricos

PU- 05: Comprobar que se devuelve la primera relación de un ciclo en un diagrama de circuito eléctrico	
Descripción	Se desea probar la función que devuelve la primera relación que forma parte de un ciclo dentro de un diagrama de circuito eléctrico.
Función	`getFirstInCycle()`
Especificación de entrada	Crear un gráfico. Seleccionar para este gráfico layout de circuito eléctrico. Añadir varios nodos conectados por relaciones formando algunas retroalimentaciones en el circuito eléctrico. Aplicar la acción de layout.

Especificación de salida	Tras la llamada a esta función se devuelve un entero siendo el identificador de la primera relación de un ciclo.

Tabla 101: PU- 05: Comprobar que se devuelve la primera relación de un ciclo en un diagrama de circuito eléctrico

PU- 06: Comprobar que se devuelve la última relación de un ciclo en un diagrama de circuito eléctrico	
Descripción	Se desea probar la función que devuelve la última relación que forma parte de un ciclo dentro de un diagrama de circuito eléctrico.
Función	`getLastInCycle()`
Especificación de entrada	Crear un gráfico. Seleccionar para este gráfico layout de circuito eléctrico. Añadir varios nodos conectados por relaciones formando algunas retroalimentaciones en el circuito eléctrico. Aplicar la acción de layout.
Especificación de salida	Tras la llamada a esta función se devuelve un entero siendo el identificador de la última relación de un ciclo.

Tabla 102: PU- 06: Comprobar que se devuelve la última relación de un ciclo en un diagrama de circuito eléctrico

PU- 07: Comprobar si un punto forma una línea con otros dos puntos	
Descripción	Se desea probar la función que comprueba que un punto de una relación, pueda formar un eje con otros dos puntos, por ejemplo, el punto que sale de un nodo al que se conecta dicha relación, y otro, siendo el punto que entra en un nodo y que también forma parte de la relación.
Función	`checkPointWithInLine(double pIniX, double pIniY, double pFinX, double pFinY, double pointX, double pointY)`
Especificación de entrada	Crear un gráfico. Seleccionar para este gráfico layout de circuito eléctrico (ya que es por ahora la única filosofía de dibujo conectada). Añadir dos artefactos conectados por una relación. Crear nuevos tipos de artefactos con representación. Modificar los artefactos con estos nuevos tipos de artefacto. Aplicar la acción de layout. Transformar el diagrama.

Especificación de salida	Tras la llamada a esta función se devuelve un booleano indicando si los tres puntos definen una línea recta, definiendo así la orientación del giro de las imágenes de los nodos asociados a los extremos de esa relación.

Tabla 103: PU- 07: Comprobar el ángulo que forman dos puntos para el giro de las imágenes

PU- 08: Comprobar el ángulo de giro de la representación de un nodo en la transformación del diagrama	
Descripción	Se desea probar la función que define el ángulo de giro de la representación de cierto nodo de un diagrama para su transformación.
Función	`rotateNodeBitmap(Node nodoAux)`
Especificación de entrada	Crear un gráfico. Seleccionar para este gráfico layout de circuito eléctrico (ya que es por ahora la única filosofía de dibujo conectada). Añadir dos artefactos conectados por una relación. Crear nuevos tipos de artefactos con representación. Modificar los artefactos con estos nuevos tipos de artefacto. Aplicar la acción de layout. Transformar el diagrama.
Especificación de salida	Tras la llamada a esta función se devuelve un double indicando el ángulo en grados que debe rotar la representación de un nodo en el modo transformado para definir la orientación de una filosofía de dibujo basada en conexión.

Tabla 104: PU- 08: Comprobar el ángulo de giro de la representación de un nodo en la transformación del diagrama

6. Planificación y presupuesto

En este punto se va a especificar la planificación que se ha realizado durante todo el proyecto junto con el presupuesto obtenido para el mismo. Este punto es muy importante, ya que permite controlar los plazos de entrega de este proyecto, pudiendo establecer los diferentes repartos de esfuerzo entre las diferentes tareas y teniendo en cuenta siempre la estimación de cuándo deben finalizarse. Además con el presupuesto se muestran los gastos que se han tenido a lo largo del proyecto, siendo este punto muy importante para justificar la eficiencia en el desarrollo e implementación del mismo.

6.1 Planificación

En este apartado se estimará el esfuerzo que se ha requerido para la consecución de este proyecto de fin de grado. Para ello no se ha empleado ningún software que automatice esta tarea, sino que se ha seguido una división mes a mes de la división del esfuerzo, dividiendo el mismo entre las tareas a realizar en este proyecto, y haciendo un seguimiento mediante Trello.

Teniendo en cuenta que el proyecto de fin de grado tiene asignados 12 créditos por aprobar dicha asignatura, y sabiendo que cada crédito son 25 horas de trabajo del estudiante según la especificación de la Universidad Carlos III de Madrid, se requerirán 300 horas para completar el proyecto. Sin embargo, y teniendo en cuenta las dificultades que surgen en cualquier proyecto de envergadura, contando con que sólo hay un desarrollador asignado al mismo, se ha decidido aplicar una deviación de 9 horas más por crédito, siendo entonces cada crédito unas 34 horas de trabajo, y considerándose este ajuste, se obtendrán **408 horas de trabajo.**

Así estas 408 horas de trabajo se destinarán al desarrollo de las diferentes tareas de mejora que se han especificado para este proyecto. Sin embargo, y debido a que este proyecto consiste en una mejora, se han designado 90 horas más al proyecto para poder realizar tareas de formación y de adaptación al proyecto original desarrollado por mis compañeros del año pasado y a las diferentes herramientas y políticas que sigue la empresa The Reuse Company para sus desarrollos. Por lo que al final se han considerado **498 horas de trabajo** para lograr terminar el proyecto.

Esta planificación surge de la necesidad de realizar una estimación que no acarree ningún tipo de problema en cuanto a fechas de entrega del mismo proyecto, lo que supondría un problema para el cliente, sobretodo en sobrecoste. Siempre, y siguiendo con la experiencia en otros proyectos realizados en empresas como SAP (para la que he trabajado), junto con la experiencia obtenida durante los 4 años que ha durado este grado, se intentará crear un “colchón” en horas que amortigüe cualquier problema encontrado en el desarrollo de este proyecto.

A continuación se realizará una división del trabajo en las diferentes tareas, junto con las horas estimadas para cada una y la fecha objetivo para terminarlas, junto con la fecha de inicio que se ha planificado para cada una.

Tarea	Horas estimadas	Fecha de inicio	Fecha de fin
Investigación y formación	90	04/02/2015	27/02/2015
Estado del Arte	50	02/03/2015	27/03/2015
Planificación	12	02/03/2015	13/03/2015
Presupuesto	5	02/03/2015	06/03/2015
Estudio de Viabilidad	35	09/03/2015	07/04/2015
Análisis	45	16/03/2015	15/04/2015
Diseño	83	08/04/2015	27/05/2015
Implementación	100	16/04/2015	12/06/2015
Pruebas	52	27/05/2015	25/06/2015
Conclusiones y líneas de mejora	26	27/05/2015	26/06/2015
Total de horas	**498 Horas**		

Tabla 105: Estimación de horas

Como se muestra en la tabla anterior, el proyecto comienza el día 4 de Febrero de 2015 terminando el día 26 de Junio de 2015, en jornadas de 5 horas, por lo que la duración del mismo será de 5 meses, dejando espacio para cualquier corrección que sea necesaria aplicar en el proyecto. A continuación se muestra el desglose de días laborales en los meses que dura este proyecto, siguiendo la estimación de 5 horas diarias.

Tarea	Días laborales	Jornada laboral	Total de horas
Febrero	18	5 horas/día	90
Marzo	21	5 horas/día	105
Abril	20	5 horas/día	100
Mayo	19	5 horas/día	95
Junio	19	5 horas/día	95
			485 Horas

Tabla 106: Horas totales por jornadas laborables en los meses trabajados

Como se muestra en el desglose de días laborables trabajados, y siguiendo una jornada de 5 horas diarias, se obtendrá un cómputo de horas de 485 horas, de tal manera que sobrarán 13 horas que en caso de subestimar el esfuerzo para cada tarea, se podrán repartir para no sobrepasar el número de horas planificadas para este proyecto.

A continuación se mostrarán el desglose de cada tarea, junto a las horas que se le ha asignado dentro de cada jornada laboral, mes a mes.

FEBRERO																		
Tarea	**4**	**5**	**6**	**9**	**10**	**11**	**12**	**13**	**16**	**17**	**18**	**19**	**20**	**23**	**24**	**25**	**26**	**27**
Investigación y formación	5	5	5	5	5	5	5	6	4	5	6	4	5	6	4	5	5	5
Estado del Arte																		
Planificación																		
Presupuesto																		
Estudio de Viabilidad																		
Análisis																		
Diseño																		
Implementación																		
Pruebas																		
Conclusiones y líneas de mejora																		

Tabla 107: Estimación de esfuerzo para el mes de febrero

MARZO																						
Tarea	**2**	**3**	**4**	**5**	**6**	**9**	**10**	**11**	**12**	**13**	**16**	**17**	**18**	**20**	**23**	**24**	**25**	**26**	**27**	**30**	**31**
Investigación y formación																					
Estado del Arte	3,0	3,0	3,0	3,0	3,0	2,0	2,0	3,0	3,0	2,0	2,0	2,0	2,0	2,0	2,5	2,5	2,5	2,5	2,5		
Planificación	1,25	2,00	0,75	1,25	1,25	1,25	1,25	1,25	1,25	1,25											
Presupuesto	1,25	1,25	1,25	1,00	1,50																
Estudio de Viabilidad						1,25	1,25	1,25	1,25	1,25	1,25	1,25	1,25	1,25	1,25	1,25	1,25	1,25	1,25	3,0	2,0
Análisis											1,25	1,25	1,25	1,25	1,25	1,25	1,25	1,25	1,25	3,0	3,0
Diseño																					
Implementación																					
Pruebas																					
Conclusiones y líneas de mejora																					

Tabla 108: Estimación de esfuerzo para el mes de marzo

ABRIL																				
Tarea	**1**	**2**	**5**	**6**	**7**	**8**	**9**	**12**	**13**	**14**	**15**	**16**	**19**	**20**	**21**	**22**	**23**	**26**	**27**	**30**
Investigación y formación																				
Estado del Arte																				
Planificación																				
Presupuesto																				
Estudio de Viabilidad	2,5	2,5	3,0	3,0	3,0															
Análisis	2,5	2,5	2,0	2,0	2,0	3,0	3,0	3,0	3,0	2,5	2,00									
Diseño						2,0	2,0	2,0	2,0	2,5	3,0	2,0	3,0	2,0	2,5	4,0	4,0	3,0	2,0	2,0
Implementación												3,0	2,0	3,0	2,0	1,0	1,0	2,0	3,0	3,0
Pruebas																				
Conclusiones y líneas de mejora																				

Tabla 109: Estimación de esfuerzo para el mes de abril

MAYO																				
Tarea	**4**	**5**	**6**	**7**	**8**	**11**	**12**	**13**	**14**	**15**	**18**	**19**	**20**	**21**	**22**	**25**	**26**	**27**	**28**	**29**
Investigación y formación																				
Estado del Arte																				
Planificación																				
Presupuesto																				
Estudio de Viabilidad																				
Análisis																				
Diseño	2,5	2,5	3,0	3,0	3,0	2,0	2,0	2,0	2,0	2,5	3,0	2,0	3,0	2,0	2,5	4,0	4,0	1,0		
Implementación	2,5	2,5	2,0	2,0	2,0	3,0	3,0	3,0	3,0	2,5	2,0	3,0	2,0	3,0	2,0	1,0	1,0	2,5	3,0	3,0
Pruebas																		1,25	1,0	1,0
Conclusiones y líneas de mejora																		1,25	1,0	1,0

Tabla 110: Estimación de esfuerzo para el mes de mayo

JUNIO																				
Tarea	**1**	**2**	**3**	**4**	**5**	**8**	**9**	**10**	**11**	**12**	**15**	**16**	**17**	**18**	**19**	**22**	**23**	**24**	**25**	**26**
Investigación y formación																				
Estado del Arte																				
Planificación																				
Presupuesto																				
Estudio de Viabilidad																				
Análisis																				
Diseño																				
Implementación	2,5	2,5	2,5	2,5	2,5	3,0	3,0	3,0	3,0	2,5										
Pruebas	1,5	1,5	1,25	1,25	1,25	1,0	1,0	1,0	1,0	1,25	2,5	2,5	2,5	2,5	2,5	2,5	2,5	2,5	2,5	
Conclusiones y líneas de mejora	1,5	1,5	1,25	1,25	1,25	1,0	1,0	1,0	1,0	1,25	2,5	2,5	2,5	2,5	2,5	2,5	2,5	2,5	2,5	2,5

Tabla 111: Estimación de esfuerzo para el mes de junio

A continuación se muestra un fragmento del diagrama de Gantt generado usando el Project como se estipulo en el apartado de gestión de proyecto.

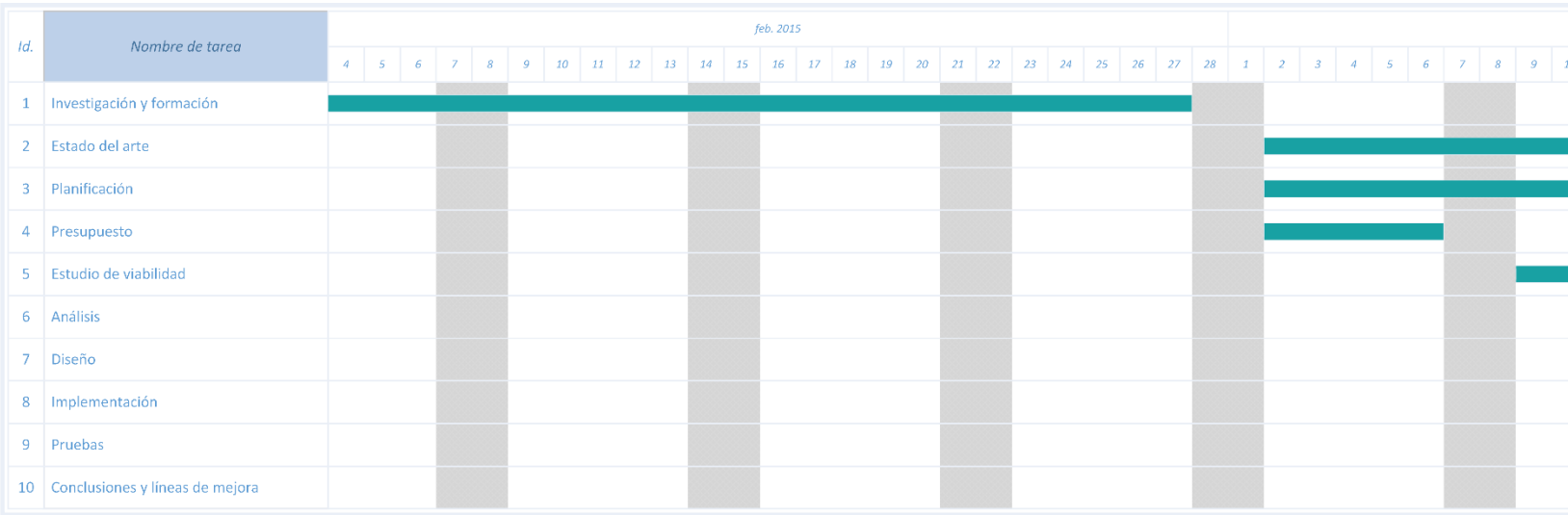

Ilustración 45: Diagrama de Gantt

6.2 Presupuesto

En este punto se recogerán los costes que se han presentado en el desarrollo de este proyecto.

Lo primero que se va a evaluar son los recursos materiales que se han utilizado a lo largo de este proyecto, así como los costes que presentan.
Para calcular estos costes se usará la métrica del % de uso, que es muy habitual para calcular los gastos materiales en todo tipo de proyectos, y que consiste en dividir el coste del material entre el número de días útiles que tiene (por ejemplo un ordenador de sobremesa tiene una vida útil de unos 4 años, dependiendo del modelo y de las prestaciones) y multiplicarlo por los días que se ha estado usando. También se incluirá en esta tabla además de los gastos de material, los costes de transporte para llegar a la oficina.

Coste Materiales/ otros elementos	Calculo del coste	Coste
Ordenador de sobremesa (con pantalla, ratón, teclado…)[2]	(Coste del pc/ Días útiles) * Días de uso durante el proyecto = (900/ 730) * 97	119,6 €
Windows 8.1	Microsoft DreamSpark for Academic Institutions	0 €
Office 2013	Microsoft DreamSpark for Academic Institutions	0 €
Visio Professional 2013	Microsoft DreamSpark for Academic Institutions	0 €
Project 2013	Microsoft DreamSpark for Academic Institutions	0 €
Visual Studio 2010 Ultimate	Microsoft DreamSpark for Academic Institutions	0 €
Licencia librería Lassalle	(Coste de la librería / Días útiles) * Días de uso durante el proyecto = (600/240) * 97	242,5 €
Ordenador portátil personal	(Coste del pc/Días útiles) * Días de uso durante el proyecto = (850/900) *40	38,8 €
Fibra Óptica Movistar	Nº de meses * precio al mes = 5 * 60	300 €
Transporte	Nº de meses * precio al mes = 5 * 60	300 €
TOTAL		1000,9 €

Tabla 112: Coste de material y transporte

[2] Todos los valores de los equipos han sido calculados a través de la aplicación configuradora de PC-COMPONENTES

En cuanto al gasto personal, hay que tener en cuenta que aunque sólo haya un desarrollador en este proyecto, este asume una buena cantidad de roles.

A continuación se desglosarán los roles que se han tenido en cuenta para la consecución de este proyecto, calculándose los mismos mediante 14 pagas anuales, los meses suelen tener entre 20 y 22 días laborales, y las jornadas son de 8 horas, junto con su sueldo bruto anual y su sueldo bruto por hora:

Rol	Salario Bruto Anual	Salario Bruto a la hora
Jefe de Proyecto	72000 €	29,2 €
Analista	40000 €	16,2 €
Programador	30000 €	12,17 €

Tabla 113: Sueldos anuales y a la hora por los roles asumidos en este proyecto.

Así aplicando el salario por hora a las horas dedicadas a cada tarea en este proyecto, se calculará el coste total de personal durante estos 5 meses.

Tarea	Jefe de Proyecto	Analista	Programador
Investigación y formación	30	30	30
Estado del Arte	10	40	0
Planificación	12	0	0
Presupuesto	5	0	0
Estudio de Viabilidad	20	15	0
Análisis	10	35	0
Diseño	23	30	30
Implementación	0	0	100
Pruebas	0	0	52
Conclusiones y líneas de mejora	10	16	0
Total de horas Por Rol	120	166	212
Coste por Rol	3504	2689,2	2580,04
TOTAL		**8773,24 €**	

Tabla 114: Coste total y por rol por esfuerzo en cada tarea

Así el coste final de este proyecto ha sido el siguiente:

Concepto	Cantidad	Total
Costes directos	8773,24 + 1000,9	9774,14 €
Costes indirectos (10 %)	977,41	10751,55 €
Margen de riesgo (10 %)	977,41	11728,96 €
Beneficio (16 %)	1563,86	13292,82 €
IVA (21 %)[3]	2791,49	16084,31 €
COSTE TOTAL	**16084,31 €**	

Tabla 115: Desglose completo del presupuesto

[3] A mes de Julio del año 2015

7. Líneas futuras

La principal mejora que consideraría para este proyecto, sería el desarrollo de más filosofías de dibujo que pudiesen aplicarse a los distintos diagramas que quiera dibujar un usuario, con el objetivo de generalizar más las representaciones que este módulo puede llevar a cabo.

Otro punto que considero interesante, sería añadir personalizaciones a los diagramas, pudiéndose configurar cualquier parámetro del mismo. Quizás fuera interesante incluso añadir sets de representaciones en forma de iconos que el usuario pudiera utilizar para cada filosofía de dibujo, del mismo modo que por ejemplo en el Visio 2013, puedes seleccionar diagramas pre-configurados con determinados set de figuras como casos de uso, diagramas ingenieriles, etc.

Así si se usaran set de imágenes junto con diferentes filosofías de dibujo, el usuario podría combinarlas para crear todos los tipos de diagramas que el conciba para cubrir sus necesidades.

Ilustración 46: Set de iconos de Visio 2013 para diagramas de redes

Yendo aún más lejos, si un usuario pudiera dibujar un diagrama de ejemplo, y la aplicación pudiese reconocer un patrón que pudiese repetir sobre este diagrama, podrían crearse todos los diagramas que se deseen. Existen ya aplicaciones que reconocen las formas de ciertos dibujos y llegan a replicarlas, por lo que su inclusión en las funcionalidades de este sistema le daría un valor inmenso.

Por último, portaría todas las aplicaciones relacionadas con el Knowledge Manager, y el mismo, a un entorno web.

8. Conclusions

8.1 Planning variances

During this project development, the planning has been successful within all the tasks assigned to the improvement of this module. Taking into account that the project´s delivery date is at mid-September, this planning give enough time to solve any problem during the project's development (It was planned to end it at the end of June, having July, August and September if it is needed).

At first there are no schedule variances in the project's planning, as previously I have taken into account to make the scheduling with enough time to take care of any delay at any development stage.

The main planning variances do not take more than 5 days, as this project has been accomplished at the beginning of July (Because of the lots of user interface tests made, and also because of the testing of the last year project's functionality).

8.2 Accomplished objectives

Although the system´s scope was limited, all the main project's objectives have been accomplished, and also the customer requirements as it can be checked on the test plan and also the acceptance tests developed during this project implementation.

Taking some extra days, I can also checked all the previous system´s functionality, and also plan the next steps of this project´s development.

8.3 Personal Evaluation

In this section, I am going to review this project in first person for the very first time.

This project has been a very difficult task, as it was the first time I develop a project as an improvement of another existing one. I have to say that I have never seen any example of the management of improvements over an existing project, so I need a long time to learn how to handle it with efficiency.

Another difficult task during the development of this project, has been searching the several algorithms used to take care of Juan Llorens' requirements. However, Juan is an experimented project manager, and he always specify me with precision all of his requirements, so I could go further with this project's development with less effort.

When you don´t need to focus so much in the customer's requirements, it's easier and more comfortable to take another tasks that will take more effort, so you can accomplish them with more time, and because of that, with a better performance.
Also, I think that this project has been satisfactory to me, as I could gain a lot of experience in project management, development… I also could see my department colleagues works, with all of their researching work, and I find that interesting.

Finally, this project has made me see the working world in a more realistic way, more than in any other moment during this bachelor.

9. Anexos

En este punto se definirán los diferentes términos y acrónimos que no se hayan definido a lo largo de la redacción de esta documentación.

9.1 Glosario de términos

- **Knowledge Manager:** Software gestor para sistemas de organización del conocimiento creado por The Reuse Company.
- **Metamodelos:** Un metamodelo establece los elementos de un lenguaje usado como modelo de un sistema, así como las relaciones entre ellos y las restricciones a las que se enfrenta el mismo.
- **Acción de verbose:** Modo en el que se despliega información adicional sobre la representación de un determinado término u artefacto (por ejemplo el tipo de artefacto asociado a un artefacto), siempre que esté no tenga asociado ningún tipo de icono.
- **Filosofía de dibujo o layout:** Hacen referencia a la disposición que reciben los elementos de un diagrama según el tipo que sea.
- **Clic:** Acción de pulsar el botón izquierdo del ratón.
- **HTML5:** lenguaje estructurado usado en la creación de páginas web.
- **Javascript:** Lenguaje de programación interpretado que es orientado a objetos y que se utiliza principalmente en el lado cliente para cargar módulos y complementos de páginas web que mejoran la interfaz del usuario notablemente. Es interpretado actualmente por todos los navegadores web que existen en el mercado.
- **Open Source:** Ya sea código, aplicaciones, sistemas operativos… Hace referencia que todos ellos no tienen una licencia de pago para su uso, sino una gratuita.

9.2 Acrónimos

- **ITIL:** Refiriéndose a la Infraestructura de Tecnologías de la información, siendo una guía usada por todas las empresa de gestión de servicios informáticos.
- **IEEE:** Instituto de Ingeniería eléctrica y electrónica, que es una asociación mundial de técnicos e ingenieros dedicada a la normalización y la creación de diferentes estándares para las nuevas tecnologías.
- **RSHIP:** Modelo de representación de la información basado en relaciones.
- **TIC:** Tecnologías de la información y la comunicación.
- **W3C**: Comunidad internacional que desarrolla los estándares para la programación web.
- **UML:** Lenguaje Unificado de Modelado (UML), define un estándar para modelar sistemas software y es uno de los más usados en todo el mundo.

9.3 Citas Bibliográficas

[1] Información acerca de serialización de imágenes en XML:
http://stackoverflow.com/questions/18841690/serialize-and-store-an-image-in-an-xml-file

[2] Visionado de imágenes desde byteArray en JavaScript:
http://stackoverflow.com/questions/9463981/displaying-byte-array-as-image-using-javascript

[3] Redimensión de imágenes en C#:
http://stackoverflow.com/questions/10839358/resize-bitmap-image

[4] Información sobre detección de ciclos I:
http://scienceblogs.com/goodmath/2007/10/30/computing-strongly-connected-c/

[5] Información sobre detección de ciclos II:
https://cw.fel.cvut.cz/wiki/_media/courses/ae4m33pal/lectures/2012pal03.pdf

[6] Definiciones de TI- ITIL:
http://itil.osiatis.es/Curso_ITIL/Gestion_Servicios_TI/fundamentos_de_la_gestion_TI/que_es_ITIL/que_es_ITIL.php

[7] Rotación de imágenes JavaScript:
http://johnveldboom.com/posts/rotate-images-using-jquery/

[8] Trabajo de fin de grado de Pablo Sánchez Pérez:[4]
https://drive.google.com/file/d/0B9N-4x2XtVvMVFVTMG8yblJRaG8/view?usp=sharing

[9] Trabajo de fin de grado de Víctor Sacristán Tejudo[4]:
https://drive.google.com/file/d/0B9N-4x2XtVvMWTg3a0U1NHY0OTQ/view?usp=sharing

[10] Trabajo de fin de grado de Alejandro Fragueiro Oliva[4]:
https://drive.google.com/file/d/0B9N-4x2XtVvMc3BrcGxpZ0wwbzQ/view?usp=sharing

[4] Estas memorias no estaban presentes en el archivo de la biblioteca, por lo que se ha habilitado un repositorio en Google Drive para poder permitir la consulta de las mismas. Todo con permiso de Juan Llorens, así como de los autores de las mismas.

9.4 User Manual (Updated notes)

9.4.1 Introduction

This manual will guide a user, in all of the improvements developed over the Graphical Visual Representation Module, part of the Knowledge Manager and also summarize all the new improvements implemented to this system. This manual will show also the ways for applying a layout to a diagram, creating new artifact types with representations, and show the diagram's nodes representations in the transformation view.

It is highly recommended to check the Knowledge Manager User manual, and this module user manual as this only describe this system's update.

9.4.2 Let's refresh your mind with the user interface

Here you can see two interfaces screenshots:

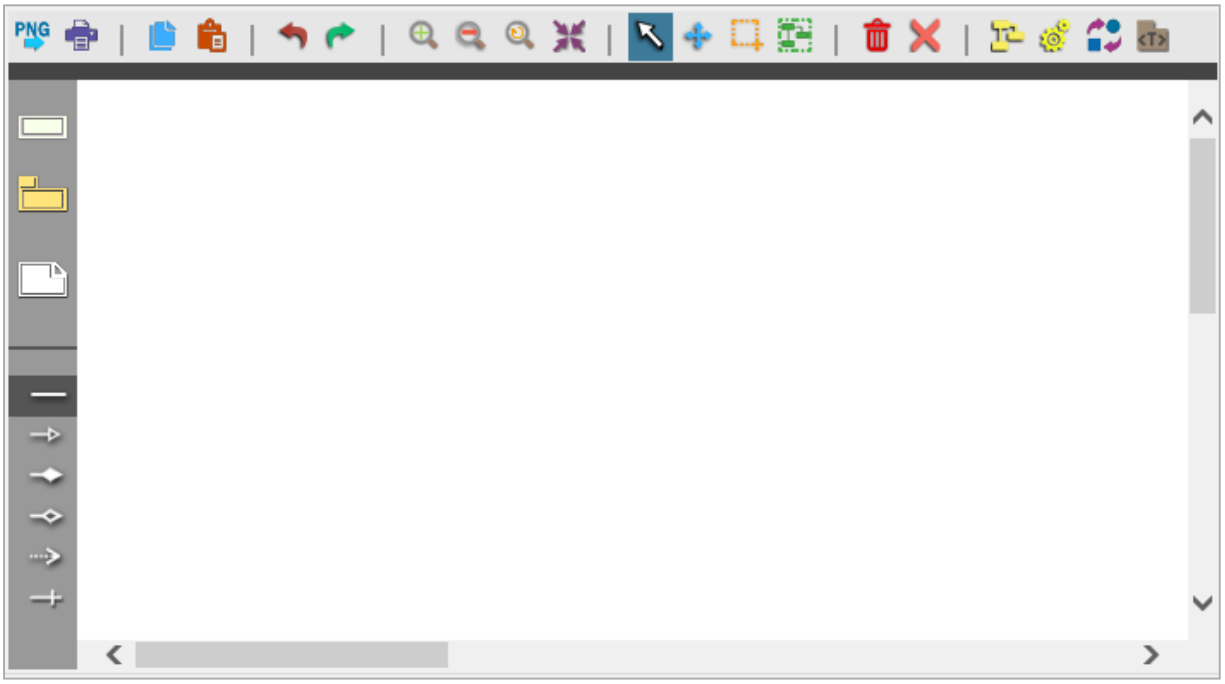

Ilustración 47: Original User Interface

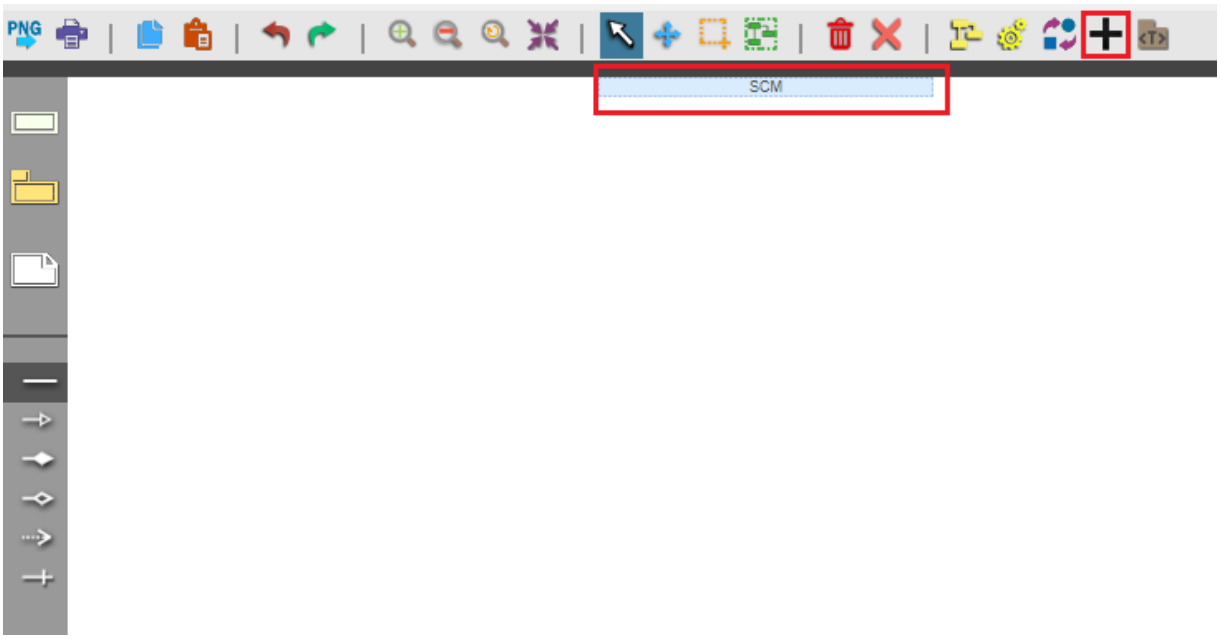

Ilustración 48: New and changed user interface

One shows the old interface and the other shows the new changes for this module version. As you can see, there are only a few changes in the interface structure.

The main differences with the older interface are:

- **Diagram´s artifact type identification box:** It is a box where a text showing the artifact type assigned to the diagram is displayed.
- **New Artifact Type button:** This button will show a pop up with a New artifact Type's form, allowing the user to create a New Artifact Type with its representation and assign a layout to this New Artifact Type if it is needed by the user. Then the user could create new layouts if he assign one of the different layouts available in the system to an Artifact Type.

All the other buttons in the toolbar and de edition's palette, give the same functionality as in the system's previous version.

It seems that there are nearly no changes with the previous version, but the next section will show and detail all the new functionality and new actions allowed for the user in this improvement.

9.4.3 Layout in the native and transformation view

Here it is intended to explain how the layout action will work in this version, as it is one of the new features of this system's improvements.

Before this update, the user could just apply a layout in the native view. Now he/she could apply also de layout in the transformation view only clicking over the layout action button, which is now displayed in the transformation view's toolbar too.

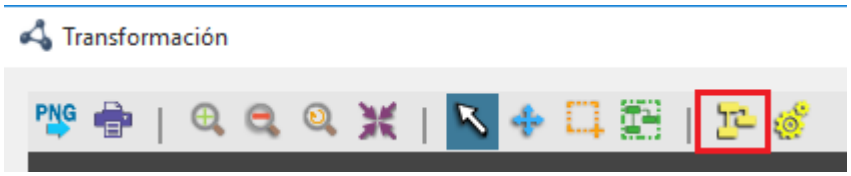

Ilustración 49: Layout action button in the transformation view

It will now apply any of the existing layouts assigned to the working diagram in the transformation view, also keeping the changes made by the user in this transformation view to the native view as is showed with the other transformation functionality improvements implemented in the next section.

9.4.4 The new artifact types with representation and the transformation view changes

In the new transformation view, there are some changes that will be detailed during this section.

- **Artifacts with Icons:**

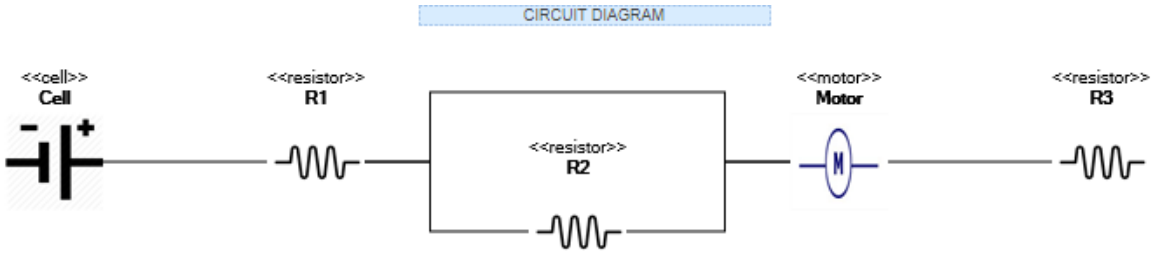

Ilustración 50: Icons in a diagram in the transformation view

As you can see in the previous screenshot, the transformation view will show icon associated to any artifact if the artifact type has a representation, and the layout allow this representations (As for example the Sequence diagram does not allow this representation). The steps to configure this kind of transformation are the following:

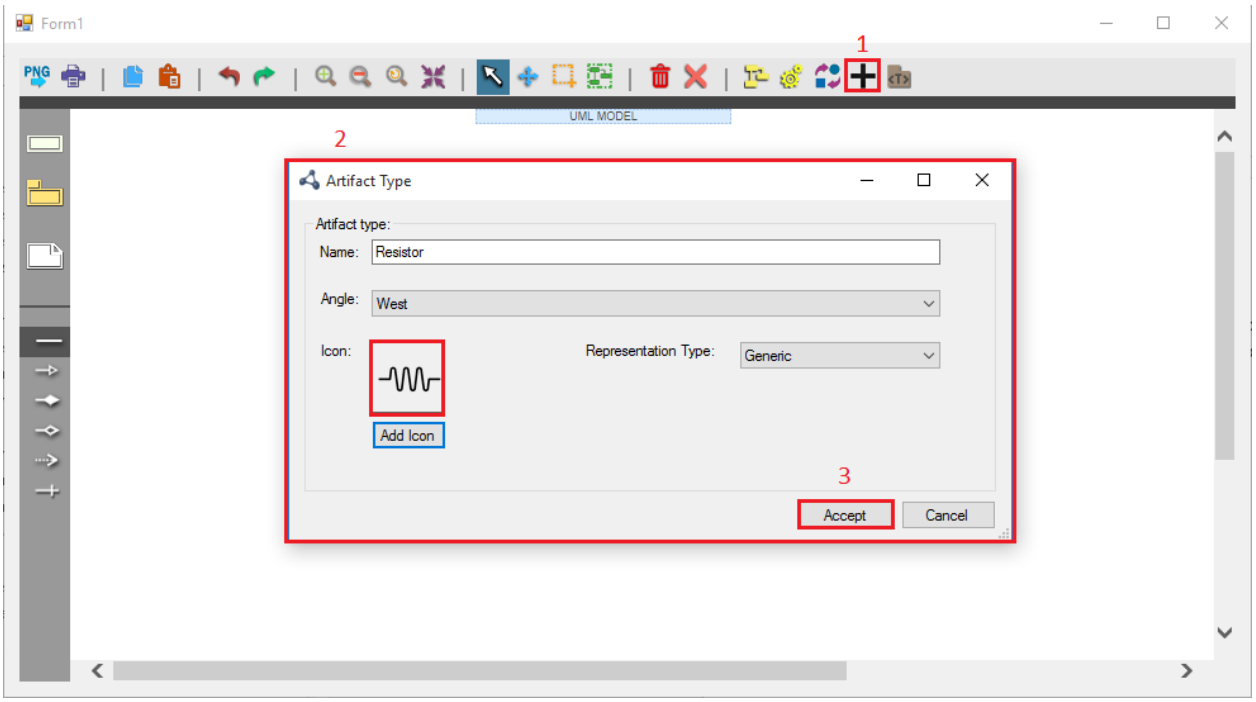

Ilustración 51: New Artifact Type Steps

- **First step:** You have to click over the New Artifact Type button, so you can add a new artifact type with a representation associated to the diagram environment.
- **Second step:** You have to fill all the form inputs for the New Artifact Type. Also the icon by selecting the add representation button, if the system can upload the selected image by the user, it will be show in the representation box (in red).
- **Third Step:** Click in Accept button to save this New Artifact Type.

Now you can use this new artifact type in any artifact of the diagram, but how can I show this icons in the transformation view? Let's see:

- **Showing artifact types icons in the transformation view:**

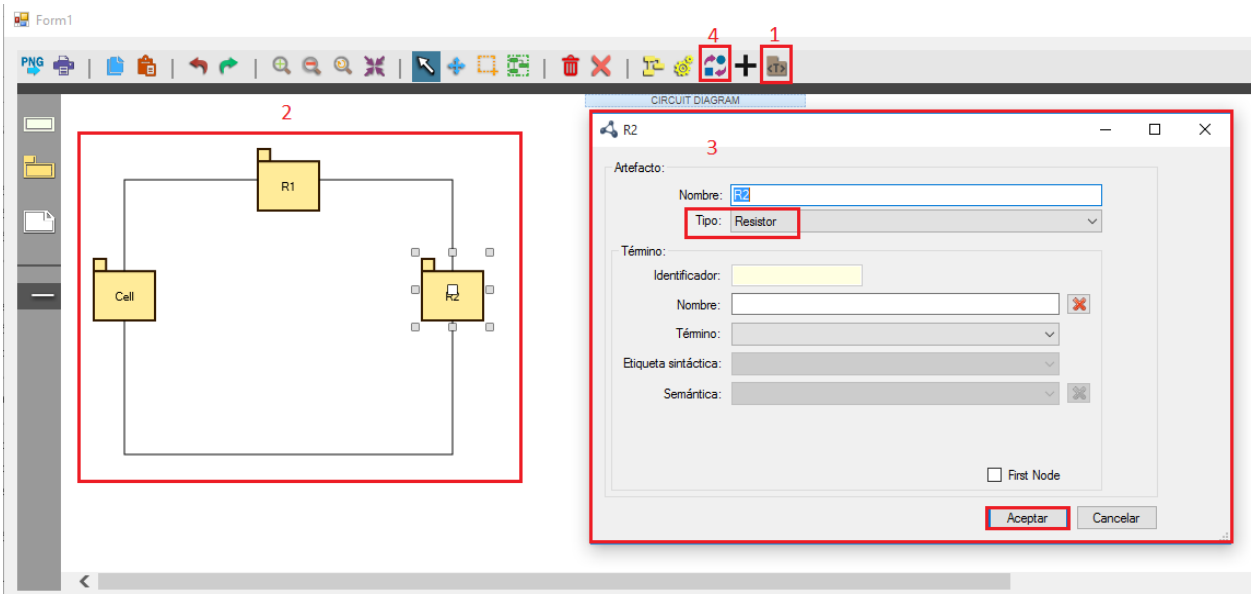

Ilustración 52: Steps in the native view to give representations to the diagram's artifacts

The steps to show these icons are the following:

- **First Step:** You have to check that the diagram's layout can show icons, for example, the circuit diagram could show icons, and rotate them so they will be in line with the other circuit's elements if this layout is applied via the layout action button. If the diagram's layout cannot show icons click over the change artifact button in the toolbar, as described in the previous version manual.
- **Second Step:** You have to add some artifacts to the diagram.
- **Third Step:** Then you have to edit them to select the previous new artifact type created by you to give the artifact this representation.
- **Fourth Step:** Click in the transformation button.

This will transform the previous diagram to this one:

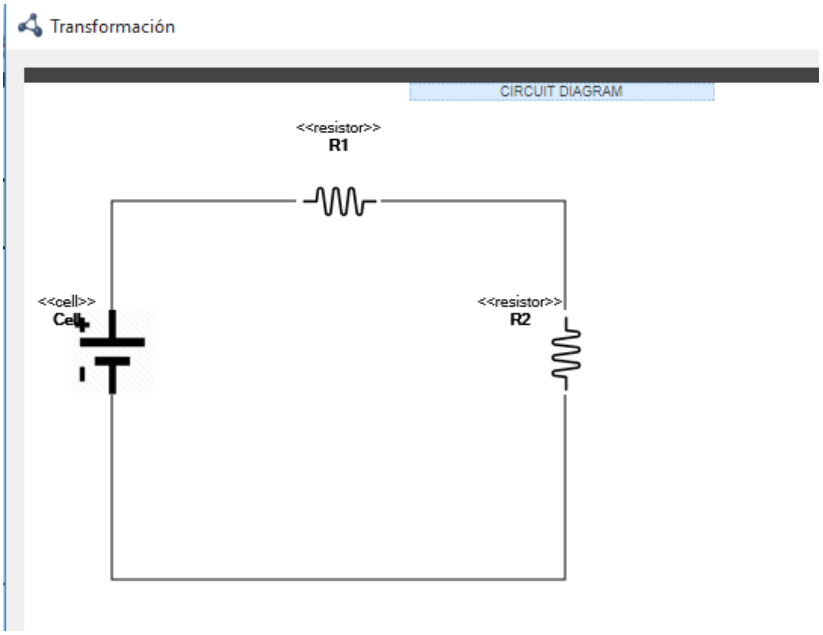

Ilustración 53: Transformed diagram with its representations

It is important to realize that any change made in the transformation view will be kept in the native view, manually made by the user, or when it is applied a layout to the diagram:

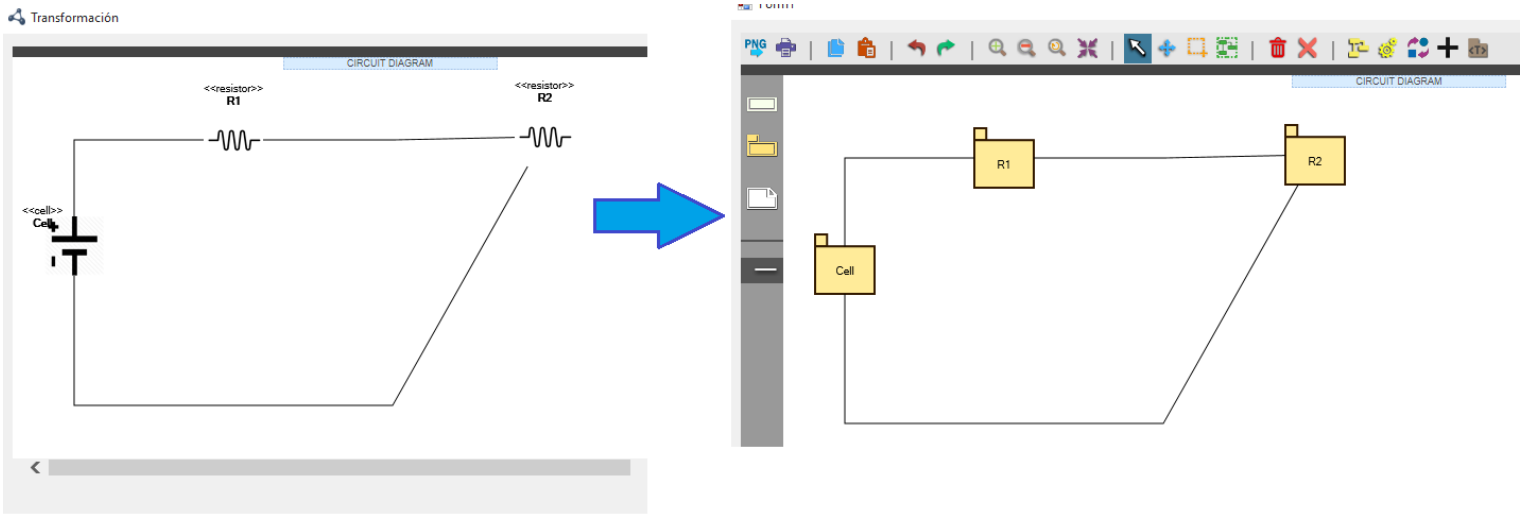

Ilustración 54: Changes kept between the native and transformation view

9.4.5 Circuit Diagram Layout

The last improvement described in this user manual would be the new layout added to the system's layout collection.

This layout is the **Circuit Diagram Layout**, which is based in algorithm to paint circuits, will organize all the elements in the diagram as a circuit, but just if the diagram could be painted as a circuit. For example, if there are two relations with the same origin and the same destination, the user will not apply this circuit diagram layout. In addition, if there are two or more relations, with the same origin and different destinations, and one of these destinations have not relations to other nodes, the user will not apply the layout.

If the user take care of these rules, he could apply this layout as described in the next steps:

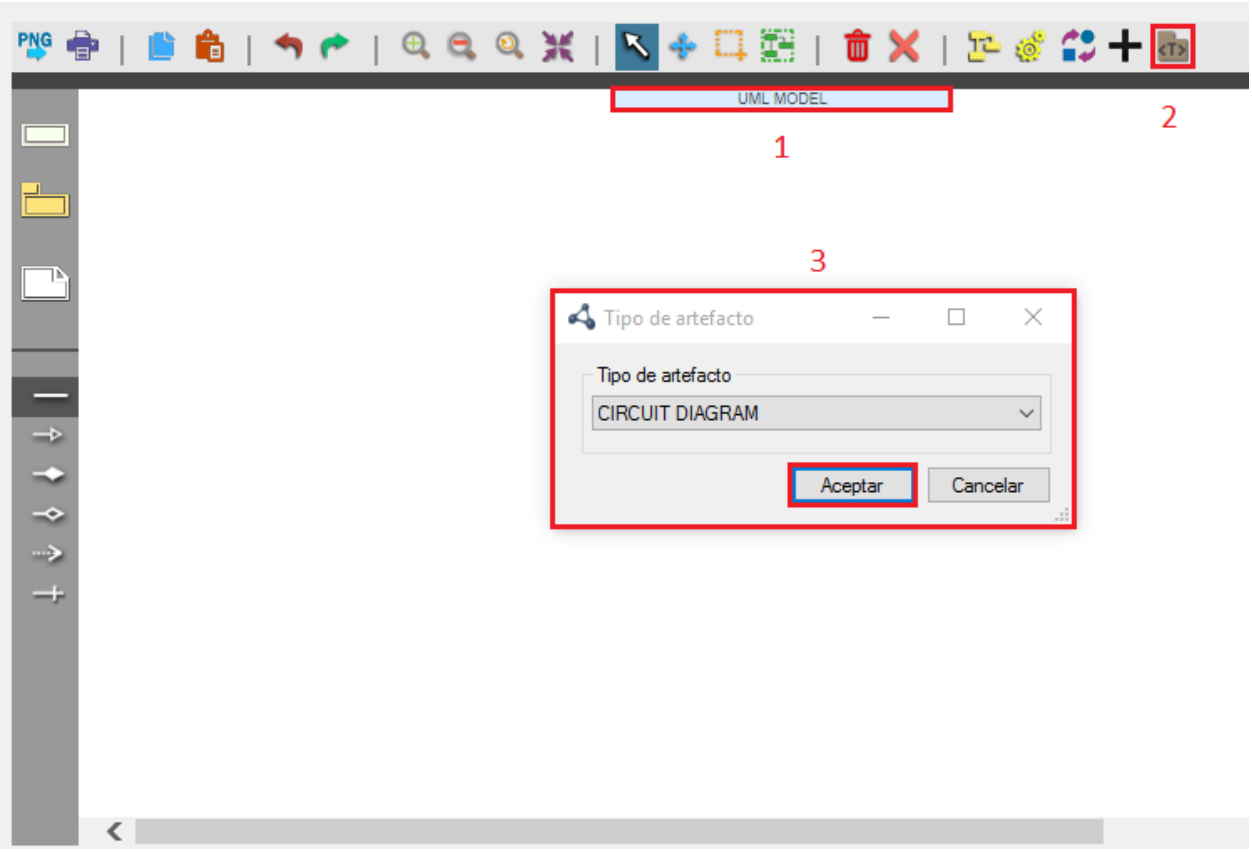

Ilustración 55: First Step to Circuit diagram action layout

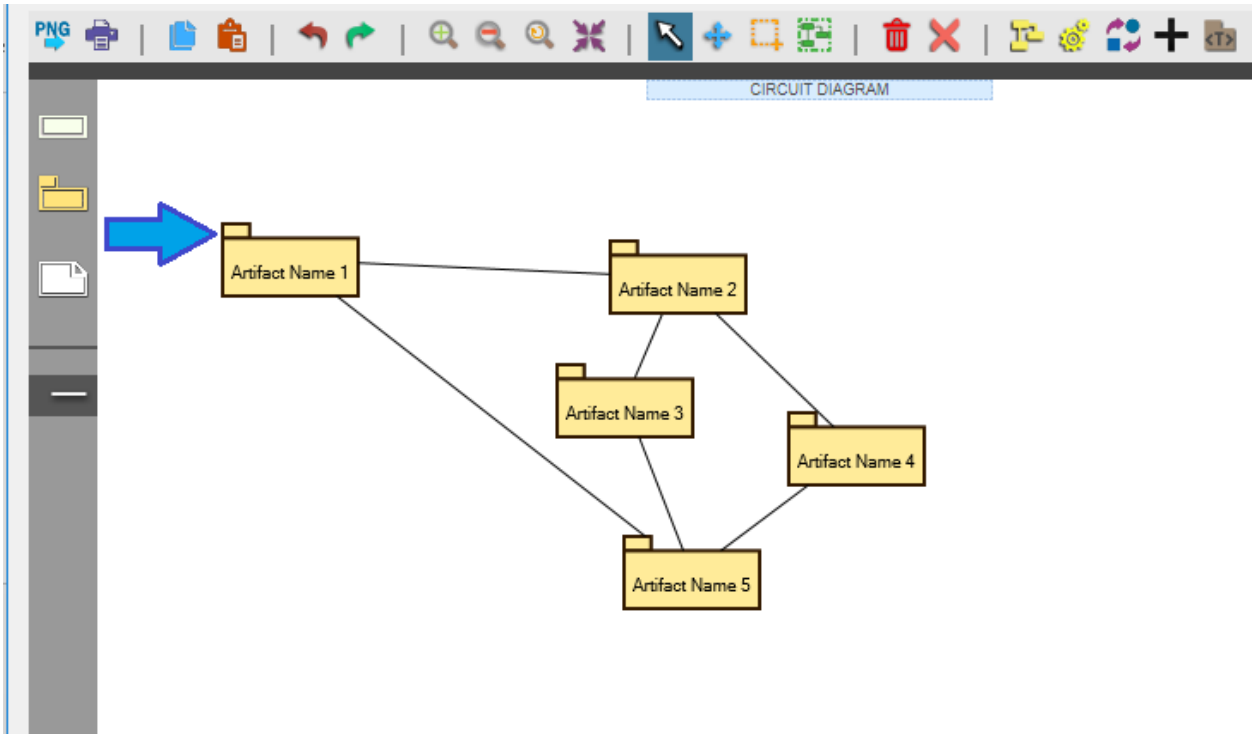

Ilustración 56: Second Step to Circuit diagram action layout

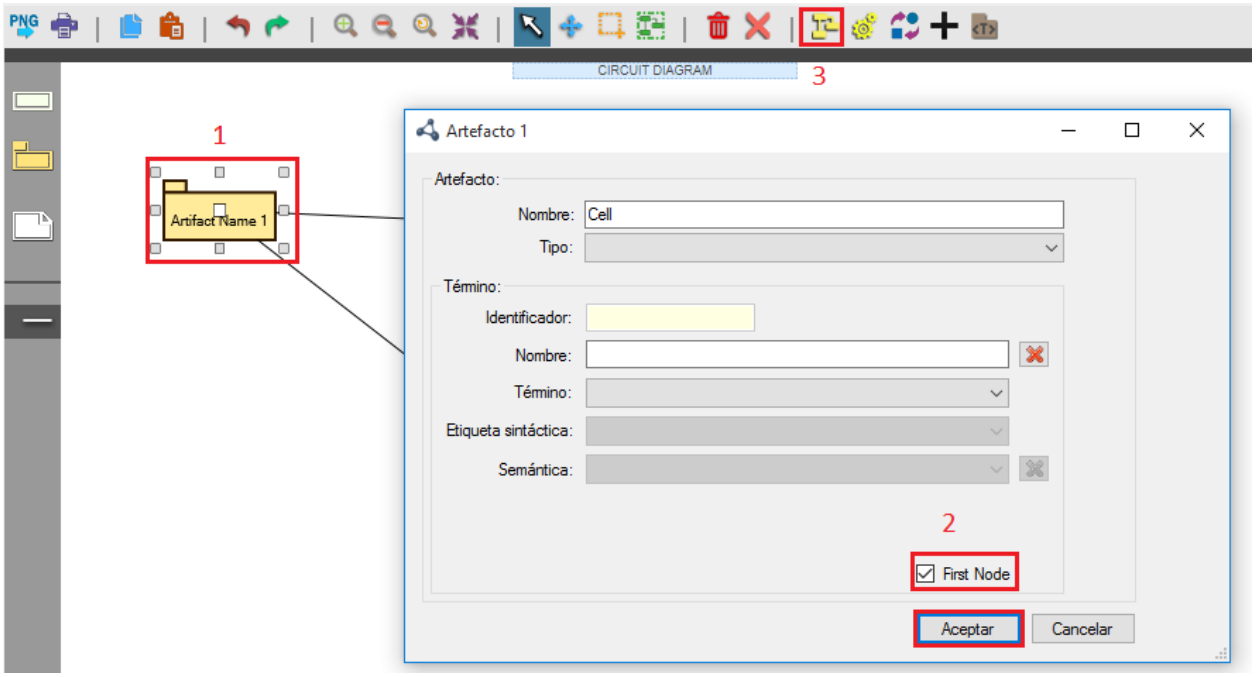

Ilustración 57: Third Step to Circuit diagram action layout

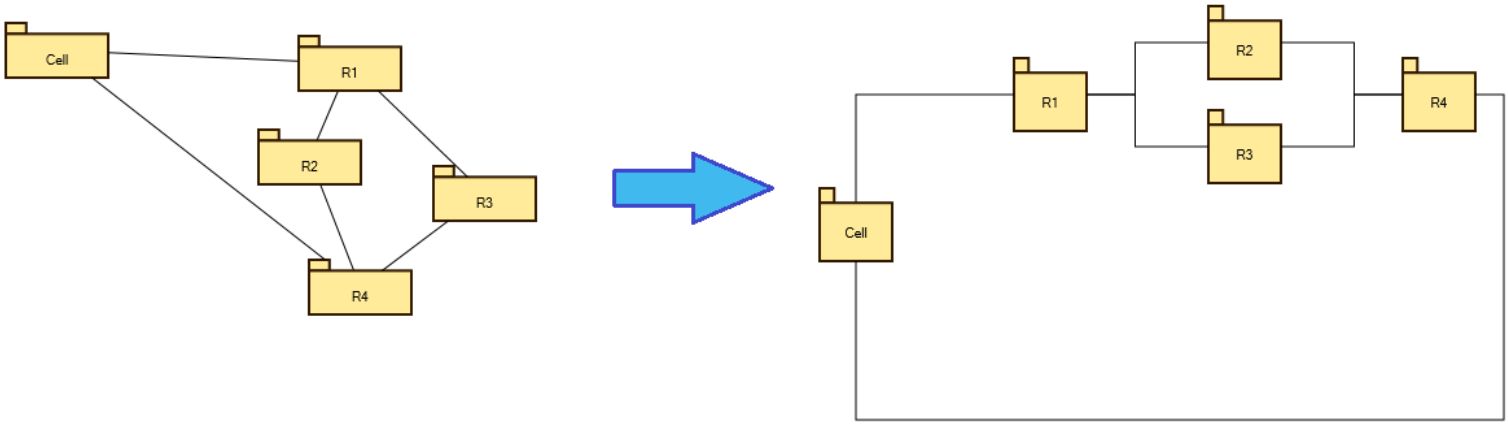

Ilustración 58: Reorganization of the nodes after applying the layout action

- **First Step:** You have to check that the diagram's layout, if it is not a circuit diagram, change it to the circuit diagram via the change artifact button.
- **Second Step:** Add some artifacts to the diagram, with some relations taking care of the circuit diagram rules.
- **Third Step:** Select if there is a first element in a cycle, and click over the layout action button.

Finally, this action could also be applied in the transformation view, and the reorganization of the nodes will be kept in the native view too.